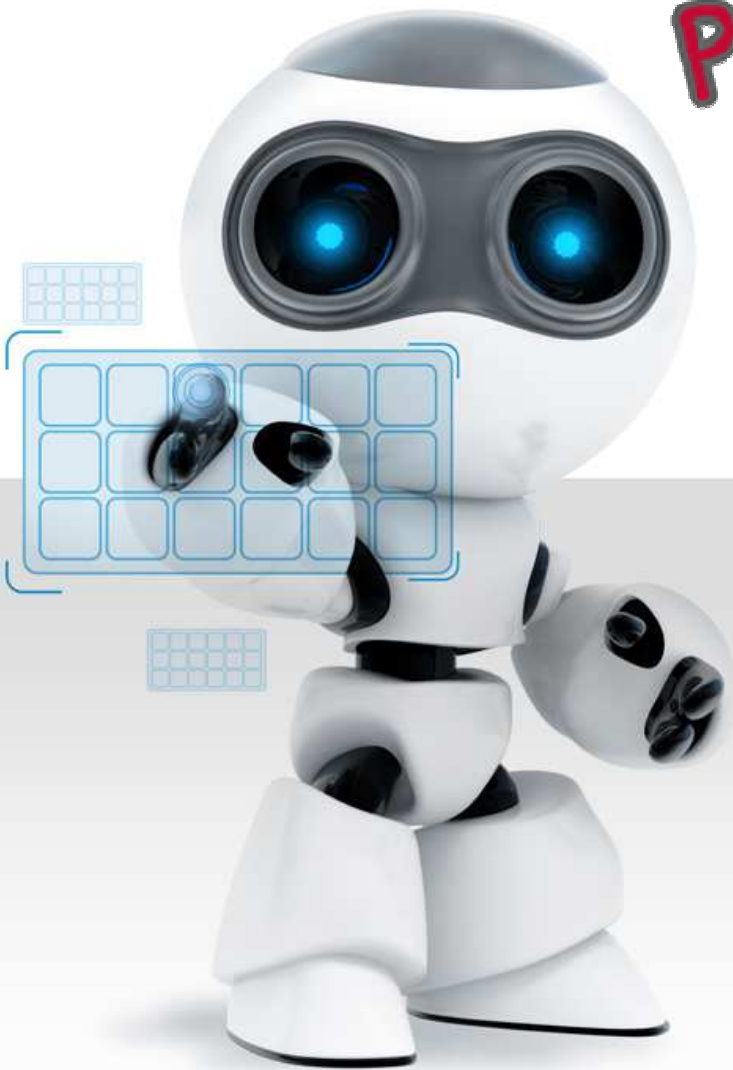


PI CODE CLUB

Edublock Tutorial

Python –Block Coding



Pi Code Club is an club specializing in scratch, Python, Raspberry PI, Robotics and block coding aiming to provide the STEM/STEAM in education, enabling young minds to get the latest in online learning education across the world. Pi Code Club currently offers Many different programs for age group 5 – 15 years

We follow a systematic sessions plan and have a well-integrated curriculum that not only encourages children to be more hands-on also promotes practical learning of STEM (Science / Technology / Engineering & Maths) using block coding and Raspberry PI Kits. Our objective is to enable school/college to introduce these programs to students in a fun & scientific style.

Edublock

Lesson 1: Lesson Plan

PI CODE CLUB

Lesson 1:

Introduction

The class will be introduced to Edublocks Python code by means of a comparison to an already familiar language, Scratch.

Learning Objectives

To understand the concepts of block based coding

To understand basic coding concepts and language

Understand how a sequence / algorithm works

Key Vocabulary

Sequence, selection and iteration.

Preparation

Subject Knowledge

An understanding of creating code in a block based environment and understanding how key coding concepts are across all languages and can be illustrated using different languages.

Possible Misconceptions

That Scratch and Python are different languages and share no concepts. This is not true. They may approach the subject differently but they both follow the same key coding concepts.

Pedagogy

The class should work in teams to complete the sequence activity (slide 5) but the remainder of the session is geared towards a teacher led session with elements of discussion.

You will need

A computer running Windows / Mac or Linux or Chromebook

A web browser (Firefox, Edge, Google Chrome, Safari)

Assessment Opportunities

Understand that coding concepts are not tied to just one language and that skills learnt in one language can be applied to others.

Basic orientation of the edublocks interface.

Understand that there are different languages for different tasks.

Outline Plan

This is a quick overview of the accompanying slide deck.

Starter activity
(Slide 3 - 5)
5 - 10 minutes

What is coding? (think/pair/share)

“What is coding?”

Ask the learners to write down what they think coding is and where it appears in their lives.

Writing code is solving a problem. We use logic to break down the problem into smaller chunks and solve these problems.

Coding concepts such as

- Sequence
 - Instructions given line by line, for example a recipe.
- Selection
 - Selecting an action based on a decision / test. For example if a driverless car sees a red traffic light, it will stop.
- Iteration
 - A loop which will run for a definite, or indefinite amount of times.

There are many different languages to write code with.

The class may be familiar with Scratch. But there are many other typed languages.

- C
- PHP
- Perl
- JavaScript
- Java

A popular language is Python and we can write code with Python following the same concepts as

<p>Group activity (Slide 6) 10 minutes</p>	<p>What is a sequence? In teams of 3 Can you guide a robot (one of the children) to draw a simple shape on a piece of paper?</p> <p>Teams are encouraged to create their own language on paper and use that to instruct the “robot”</p> <p>Suggestions for commands are</p> <ul style="list-style-type: none"> ● Forward ● Backward ● Left ● Right ● Spin left ● Spin right <p>Some children may already be familiar with Scratch and so they may use Scratch block commands, and coding concepts to automate drawing the shape (loops)</p> <p>Reward any teams that manage to retrieve the gold star.</p> <p>Extension Activity Debugging Teams should swap their code with the team next to them, can their robot follow the instructions?</p>
--	--

<p>10 minutes Slides (7 - 13)</p>	<p>How do we move from blocks to Python?</p> <p>Introduce edublocks.org website and the various modes available.</p> <p>We are focusing on the Python mode.</p> <p>Give the class an overview of how edublocks works.</p> <p>Blocks connect together just like Scratch. We are writing Python code using these blocks. Compare blocks such as while True: for i</p> <p>Compare how a block language builds up code, to how Python code is written line by line.</p>
<p>Plenary 5 minutes (Slide 14)</p>	<p>What have we learnt?</p> <ul style="list-style-type: none"> ● Coding concepts are shared across many different languages. ● If we learn the coding concepts then we can use other languages more easily. ● Block languages are ideal for introducing coding. ● Computers require clear instructions to work correctly. ● There are many typed languages used in different industries for different purposes.

<p>Next time 5 minutes (Slide 15)</p>	<p>Class will get hands on with edublocks and learn how to write Python code using blocks.</p> <p>Class will create a sequence of code to control a drawing tool called “Turtle”</p>
---	--

PI CODE CLUB



Lesson 1

Introduction

By the end of this lesson, you will...

- Understand the concepts of block-based coding
- Understand basic coding concepts and language
- Understand how a sequence/algorithm works

PI CODE CLUB

What is “coding”?

What do you think “coding” means?

Write down what coding means to you and where it appears in your life.



Coding == Problem Solving

Writing code is a tool to solve problems!

For example, Amazon Alexa is written using code and it provides a way to entertain, shop and talk to others.

Alexa is a large project! Too big for one person to work on. The project is broken down into smaller projects, and teams work on code to solve smaller problems.



Coding Concepts

Every coding language uses the same key coding concepts!

Sequence

The instructions for our code

Selection

Using logical tests to change the flow of the sequence

Iteration

Using loops to repeat sequences of code

PI CODE CLUB

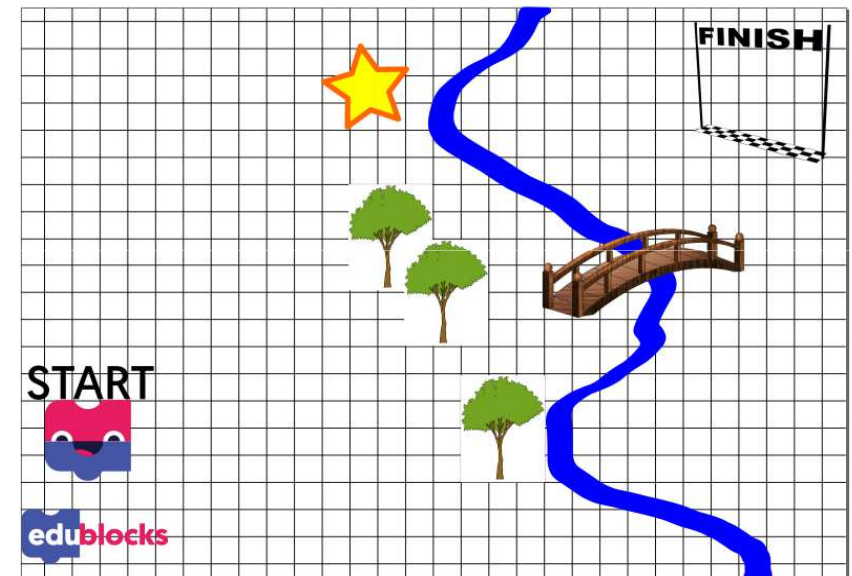
Writing our own language

Can you help Ed get to the finish line?

In teams, create a written language to guide Ed from start to finish.

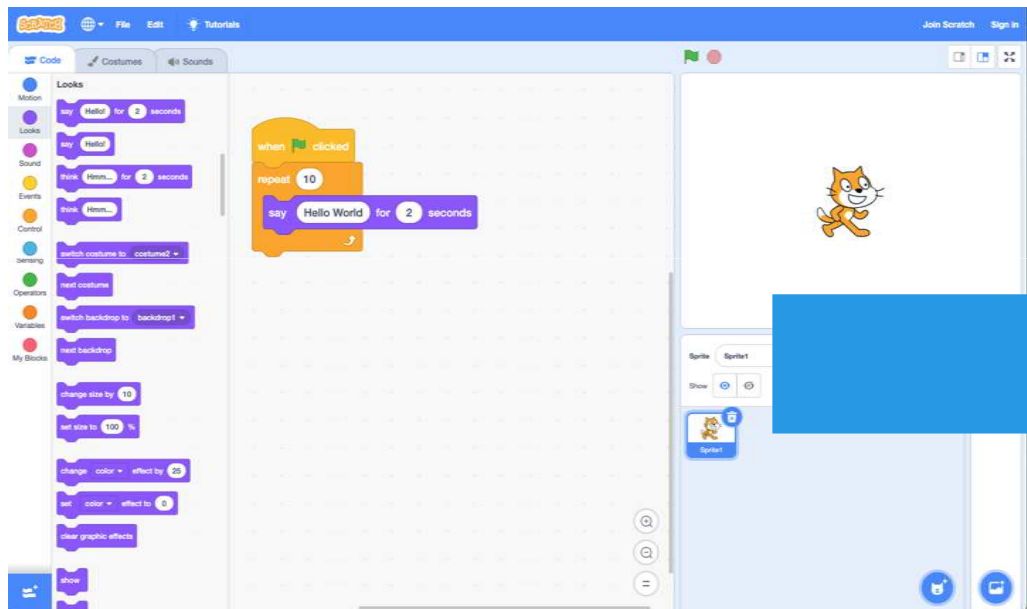
He cannot touch trees and cannot walk through water.

Your language should be easy to understand as Ed is easily confused.



Collect the star for bonus points!

Moving from Scratch to Python



Code

```
1 #Start code here
2 import time
3 for i in range(10):
4     print("Hello World")
5     time.sleep(1)
```

PI CODE CLUB

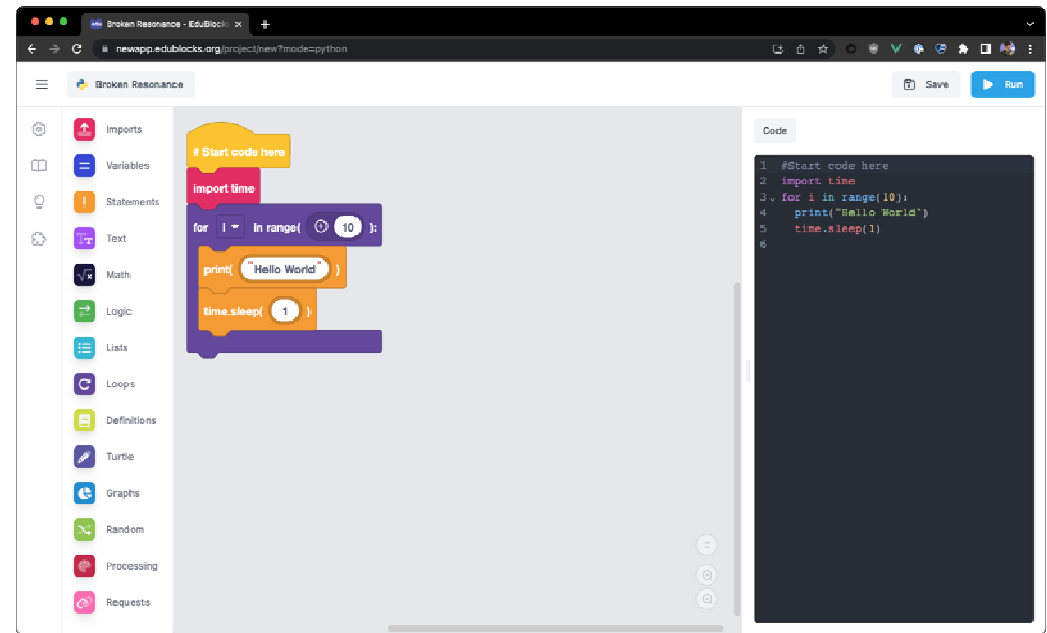
What is EduBlocks?

EduBlocks is a free application available online at edublocks.org

It was started by a 12 year-old student who wanted to help others learn how to code.

It runs in the browser like Scratch and works on all operating systems

We can write code in pure **Python** or **HTML**, or code for use with the micro:bit, Raspberry Pi and Adafruit CircuitPython controller boards.



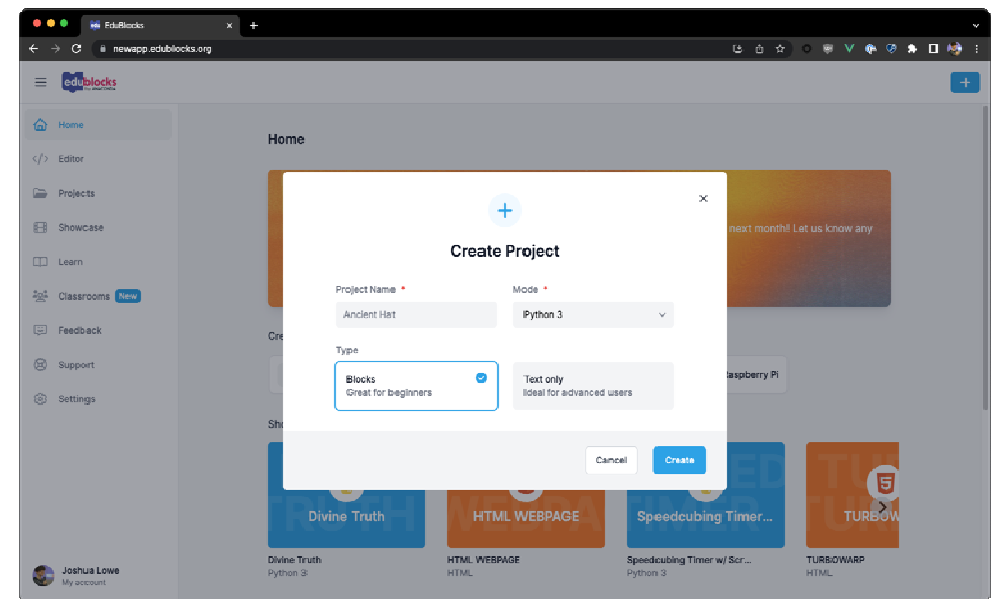
Using modes

To write code, we'll need to create a project.

Today we'll be using Python 3, so we can leave all the default options and click **"Create"**

To access the different modes, we can go to:

app.edublocks.org/editor



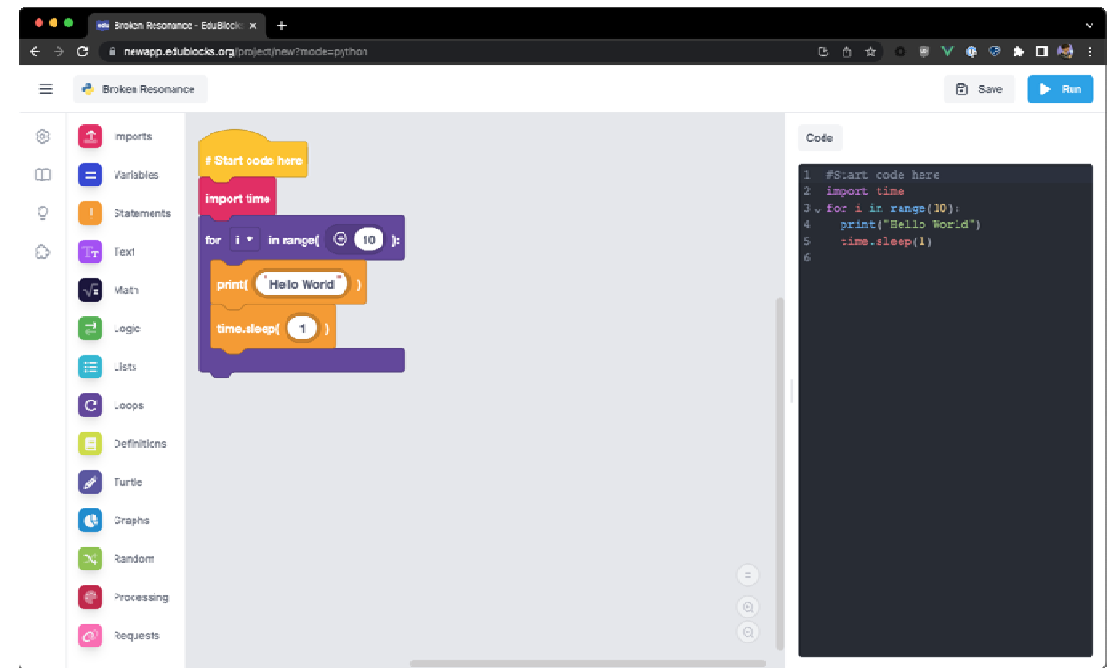
The Interface

The EduBlocks Interface is simple

On the left we have all of the blocks we can use to write code.

The blocks are places in the coding area on the left of the screen.

On the right we can see the generated Python code.



Building a sequence of code

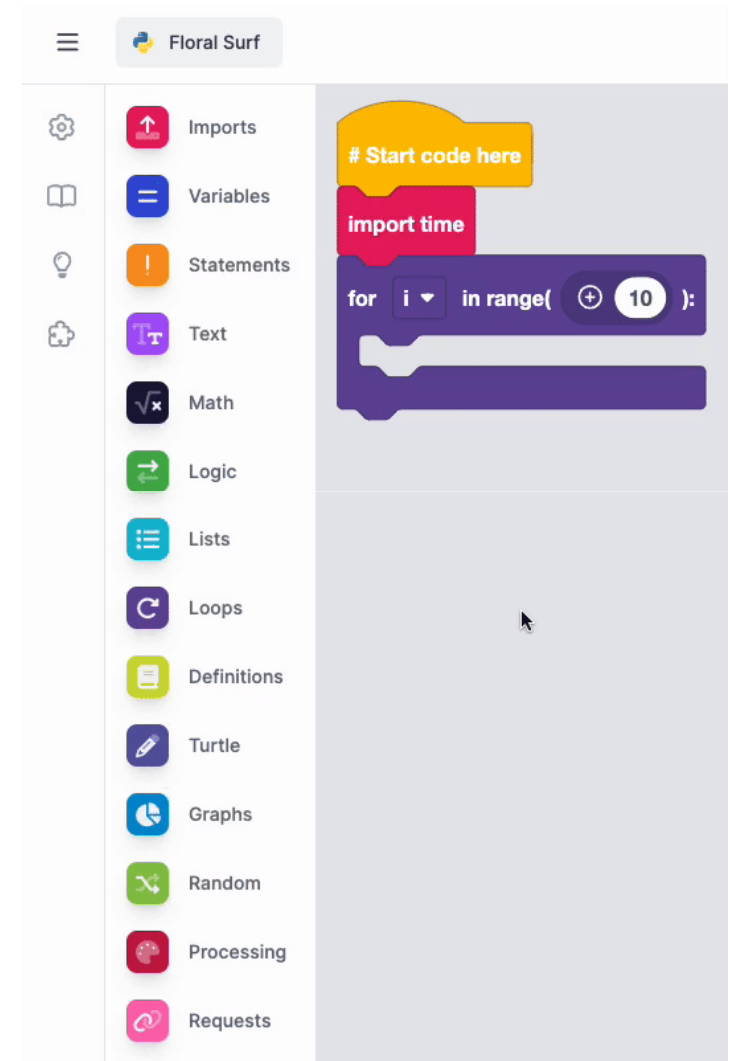
We write Python code using the blocks on the left.

The blocks click together like Scratch.

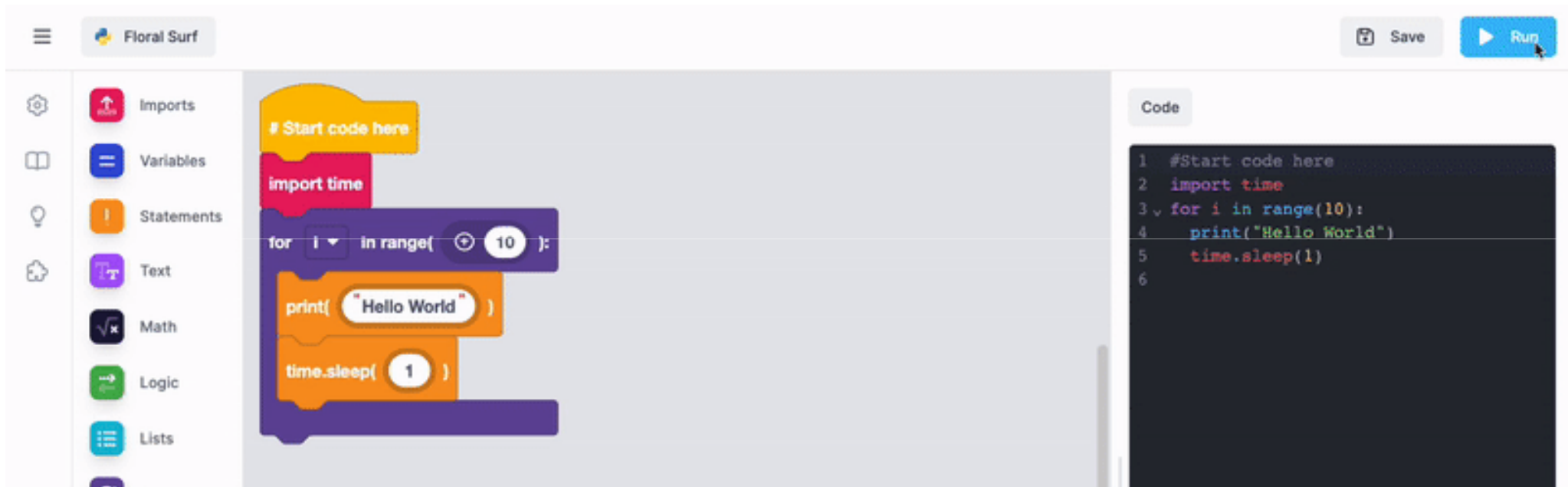
Blocks can be taken apart and put back together.

Blocks are organised in the menu by their function.

PI CODE CLUB



Run the code



The screenshot shows the edublocks code editor interface. On the left is a sidebar with a menu and categories: Imports, Variables, Statements, Text, Math, Logic, and Lists. The main workspace is divided into two views: a block-based view on the left and a text-based view on the right. The block view shows a script starting with a yellow comment block '# Start code here', followed by a red 'import time' block, a purple 'for' loop block with 'i' as the variable and '10' as the range, containing an orange 'print("Hello World")' block and an orange 'time.sleep(1)' block. The text view on the right shows the equivalent Python code:

```
1 #Start code here
2 import time
3 for i in range(10):
4     print("Hello World")
5     time.sleep(1)
6
```

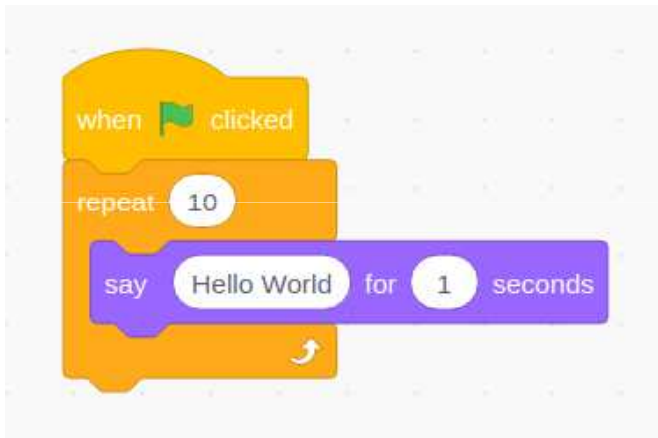
At the top right of the editor, there are 'Save' and 'Run' buttons. The 'Run' button is highlighted with a mouse cursor.

To run our code we click the Blue run button. This shows the output on the right of the screen.

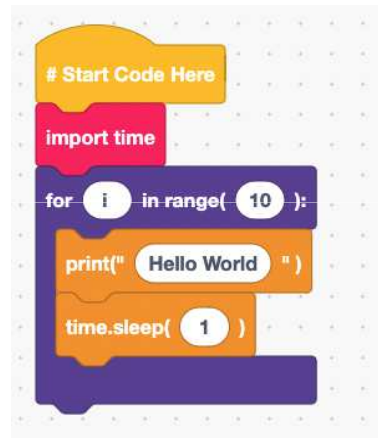
PI CODE CLUB

Same concept, different language

Scratch



EduBlocks



Python

```
1 # Start Code Here
2 import time
3 for i in range(10):
4     print("Hello World")
5     time.sleep(1)
```

Each of these code snippets will print Hello World on the screen 10 times. The same coding concept, a loop which iterates 10 times, is used in each snippet.

PI CODE CLUB

What have we learnt?

- Coding concepts are shared across many different languages.
- Block languages are ideal for introducing coding.
- Computers require clear instructions to work correctly.

PI CODE CLUB

Next Lesson

Next lesson, we will use
EduBlocks to create graphics
with Turtle

PI CODE CLUB

Edublock

Lesson 2: Lesson Plan

PI CODE CLUB

Lesson 2:

Introduction

The class will be introduced to Edublocks Python code by creating a sequence of code that will draw shapes and patterns on the screen.

Learning Objectives

How to use edublocks

To understand how a sequence of code works.

How to use different loops to repeat a sequence.

Key Vocabulary

Sequence, selection and iteration, modules.

Preparation

Subject Knowledge

An understanding of creating code in a block based environment and understanding how key coding concepts are across all languages and can be illustrated using different languages.

Possible Misconceptions

That block languages are not as powerful as typed languages.

Pedagogy

Ideally the class will each have access to a computer and complete the tasks individually. The lesson can be completed with 1 computer per 2 children.

You will need

A computer running Windows / Mac or Linux or Chromebook

A web browser (Firefox, Edge, Google Chrome, Safari)

Assessment Opportunities

Understand that coding concepts are not tied to just one language and that skills learnt in one language can be applied to others.

Basic orientation of the edublocks interface.

Outline Plan

This is a quick overview of the accompanying slide deck.

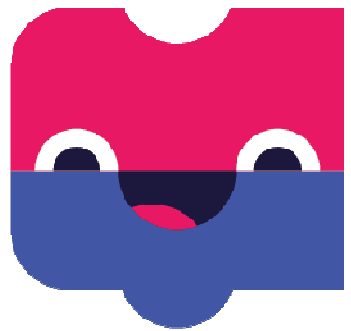
<p>Introduction (Slide 3 - 4) 5 Minutes</p>	<p>Turtle is a Python module (a pre-written library of code that we can use in our code) that enables learners to draw images on the screen.</p> <p>Turtle was created as part of the Logo language in 1967 as a means to learn how a sequence of code works. It used a robot connected to the computer to draw patterns on paper.</p> <ul style="list-style-type: none"> • With Turtle we can create a sequence of code in Python and see the output as an image. • Using different loops to repeat sections of the sequence. • Using maths and angles to draw shapes using simple equations.
<p>(Slide 5) 5 Minutes</p>	<p>To start Edublocks, ask the class to open a web browser and type in</p> <p>app.edublocks.org/editor</p> <p>Select Python 3 as the mode, give the project a file name and click Create</p> <p>Now is a good time to remind the class on how the edublocks interface works.</p> <p>To the left are the blocks, organised by function.</p> <p>Blocks are dragged from the left, into the centre of the screen and this is where we build our sequence.</p> <p>In the top right is the RUN button which will run the code.</p>

<p>5 Minutes Slide (6)</p>	<p>To use Turtle we first import the Turtle module, found in the Turtle section of blocks.</p> <ul style="list-style-type: none"> • Drag the import Turtle block into the coding area. • Then we drag the turtle = Turtle() block and connect it to the previous block. This block enables the learner to quickly use the Turtle module without using the full syntax to call a class. This enables the class to get started much faster. • We then drag the screen = Screen() block and attach it to the previous. This tells Python that we wish to draw on a section of the screen.
<p>10 Minutes (Slide 7)</p>	<p>Here the class is challenged to draw a square using Turtle.</p> <p>Turtle has blocks to move around the screen. turtle.forward(90) will move the turtle 90 pixels (steps) forward. This block has a drop down selection to move left, right and backward; these options can be used to rotate the turtle, or move backwards.</p> <p>By default the turtle will face to the right of the screen. So moving forward will move the turtle 90 pixels to the right. If we use turtle.left(90) then the turtle will spin 90 degrees anti clockwise, pointing the turtle upwards.</p> <p>The class need to understand the basic movement concepts, and use them to draw a square. Every learner should achieve this before moving on.</p> <p>All of the blocks on the slide can make a square, the learner just need to duplicate a few to finish the task.</p>

<p>Group Activity 5 Minutes (Slide 8)</p>	<p>Using a loop with a definite iteration, a for loop. We can iterate the loop four times to perform two steps that will move the turtle forwards, and rotate 90 degrees.</p> <p>The class should understand that a definite iteration loop, a for loop, is the best option to repeat a sequence of code for a set number of times.</p>
<p>Group Activity 5 Minutes (Slide 9)</p>	<p>Here we reuse the definite iteration loop from slide 8. But this time we use a calculation to determine the exterior angle required to draw a specific shape.</p> <p>For example a square has four sides so to calculate the turn angle we use</p> $360 / 4 = 90$ <p>For a triangle the class need to work out the angle by altering the calculation.</p> $360 / 3 = 120$ <p>Replacing the ?? with 120 will draw a triangle on the screen.</p> <p>Please note that Turtle uses the exterior angle to control the direction in which it faces. Some learners may expect to use interior angles.</p>
<p>5 Minutes (Slides 10-11)</p>	<p>Using two different loops. while True: which runs the pattern code forever, and for i in range() which will draw the triangle shape.</p>

<p>Group Activity 10 Minutes (Slides 12-14)</p>	<p>Slide 12 introduces changing the colour of the pen which is used to draw upon the screen.</p> <p>Colours are a mix of Red, Green and Blue (RGB) and they are given values from 0 to 255. With 0 meaning none of that colour, and 255 meaning all of the colour. So red would be 255,0,0. The width of the pen can also be changed to create thick / thin lines.</p> <p>Slide 13 is a challenge to the learners with criteria on which they can be assessed. All of the code to draw a pattern has been shown in the class.</p> <p>Slide 14 is there as an example to the class. If some get stuck, or require clarification this slide can be used to illustrate the concept.</p>
<p>Group Activity 5 Minutes (Slide 15)</p>	<p>Turtle has a special block, turtle.circle() which is a quick and easy way to draw a circle. The value turtle.circle(50) means that a circle with a radius of 50 pixels will be drawn on the screen.</p> <p>Can the class use this to draw a repeating pattern?</p>
<p>Plenary 2 Minutes (Slide 16)</p>	<p>Here we recap the learning from this lesson.</p> <ul style="list-style-type: none"> ● How to control Turtle and write a sequence of code. ● How we used different loops to perform certain actions (drawing triangles and repeating the code indefinitely) ● We learnt how to calculate angles necessary to draw a certain shape.
<p>Next Time 1 Minute (Slide 17)</p>	<p>We will continue our learning with edublocks, and learn how to capture data from a user, and use it to control the code.</p>

PI CODE CLUB



Lesson 2

What is Turtle?

PI CODE CLUB



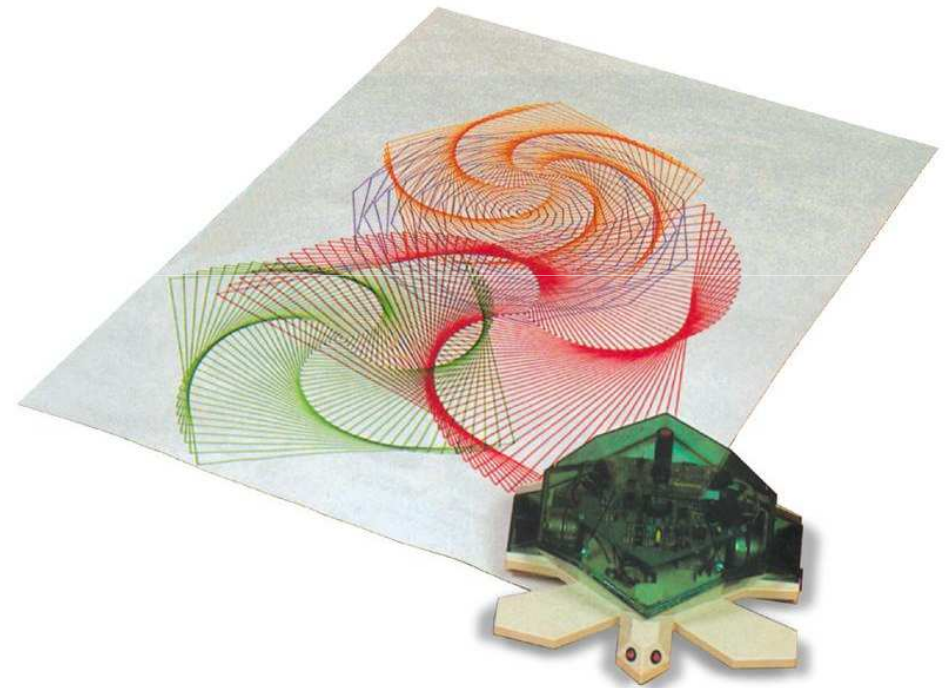
By the end of this lesson, you will...

- Understand how to use EduBlocks
- Understand how a sequence of code works
- Understand how to use different loops to repeat a sequence

What is Turtle?

Turtle is a way to draw with Python

Turtle was introduced in 1967 via a language called "Logo"



Turtle is a great introduction

Remember the coding concepts?

Using Turtle we can understand:

- How a sequence of code works
- How to use loops to repeat a sequence
- How to calculate angles to draw shapes

```
# Start code here
from turtle import *
turtle = Turtle()
screen = Screen()
screen .bgcolor( "black" )
turtle .speed( 0 )
colours = [ "red", "purple", "blue", "green", "orange", "yellow" ]
for i in range( 360 ):
    turtle .color( colours [ i % 6 ] )
    turtle .width( i / 100 + 1 )
    turtle .forward ( i )
    turtle .left ( 59 )
```

Getting Started

Open a browser

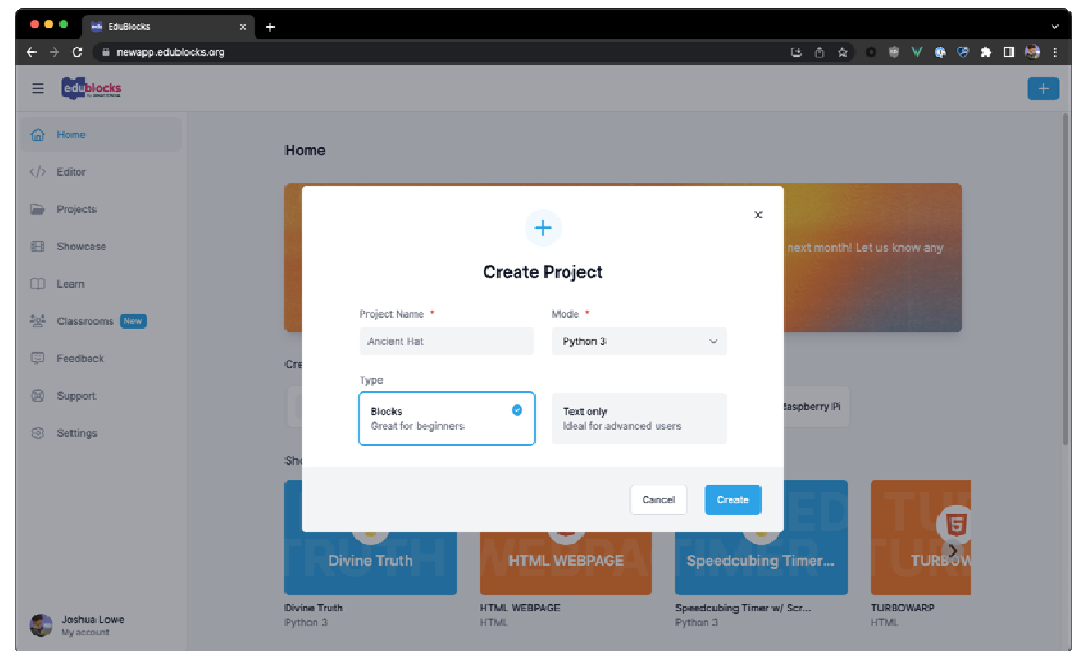
Go to app.edublocks.org/editor

Select a mode

We'll be using Python, so select "Python 3"

Get started!

Rename your project to "Lesson 2" and click **Create!**



Using Turtle

To use Turtle we need to find the blocks

In order to use Turtle we need to import it, this can be found in


Imports

All the blocks for using Turtle can be found under the **Turtle** section.

The screenshot displays the Edublocks interface. At the top, there are two pink buttons: 'Imports' with an upward arrow icon and 'from turtle import *'. Below these, a vertical sidebar on the left lists various block categories: Imports (pink), Variables (blue), Statements (orange), Text (purple), Math (dark blue), Logic (green), Lists (light blue), Loops (purple), Definitions (light green), and Turtle (dark blue). The main workspace on the right is titled 'Turtle' and contains several blue blocks. The first block is 'turtle = Turtle()'. The second block is 'screen = Screen()'. The third block is 'screen .setup(1920 , 1080)'. The fourth block is 'screen .bgcolor(255 , 0 , 0)'. The fifth block is 'screen .colormode(255 , 0 , 0)'.

Drawing a shape

Create the code on the right

Click  to see what happens.

Can you finish the code to draw a square?

```
# Start code here
from turtle import *
turtle = Turtle()
screen = Screen()
turtle . forward ( 90 )
turtle . left ( 90 )
turtle . forward ( 90 )
turtle . left ( 90 )
```

Can we improve the code?

Using a for loop, we can see the number of times that our loop will iterate

This type of loop has definite iteration

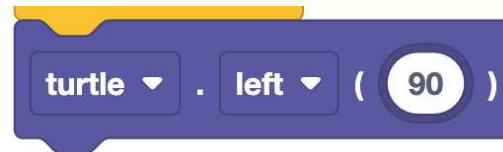
Have you noticed the left turn value is 90?

What does that mean?

```
# Start code here
from turtle import *
turtle = Turtle()
screen = Screen()
for i in range(4):
    turtle.forward(90)
    turtle.left(90)
```


Can we draw a triangle?

The turning angle

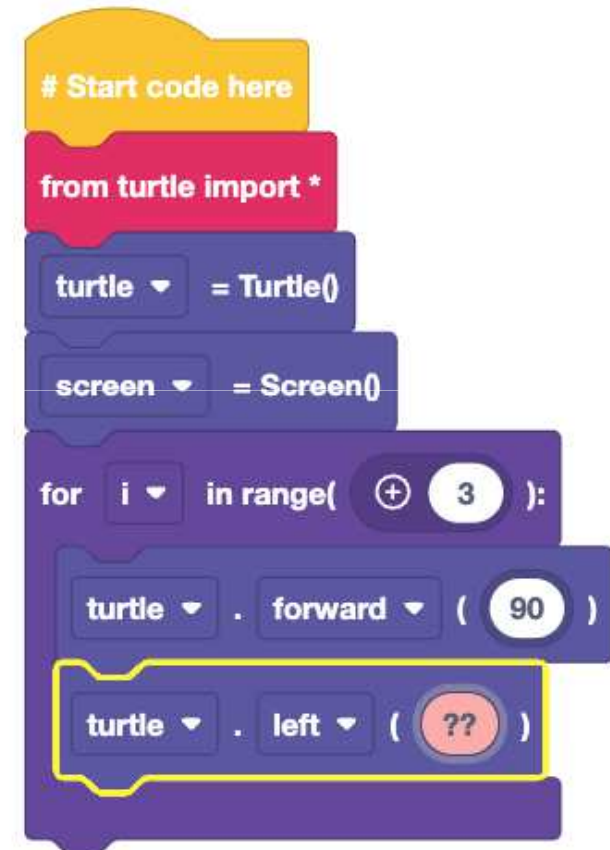


Is set by dividing 360 by the number of sides a shape has

For example a square has 4 sides

$$360/4 = 90$$

So, how do we draw a triangle?



Drawing patterns

Patterns are repeating sequences of code

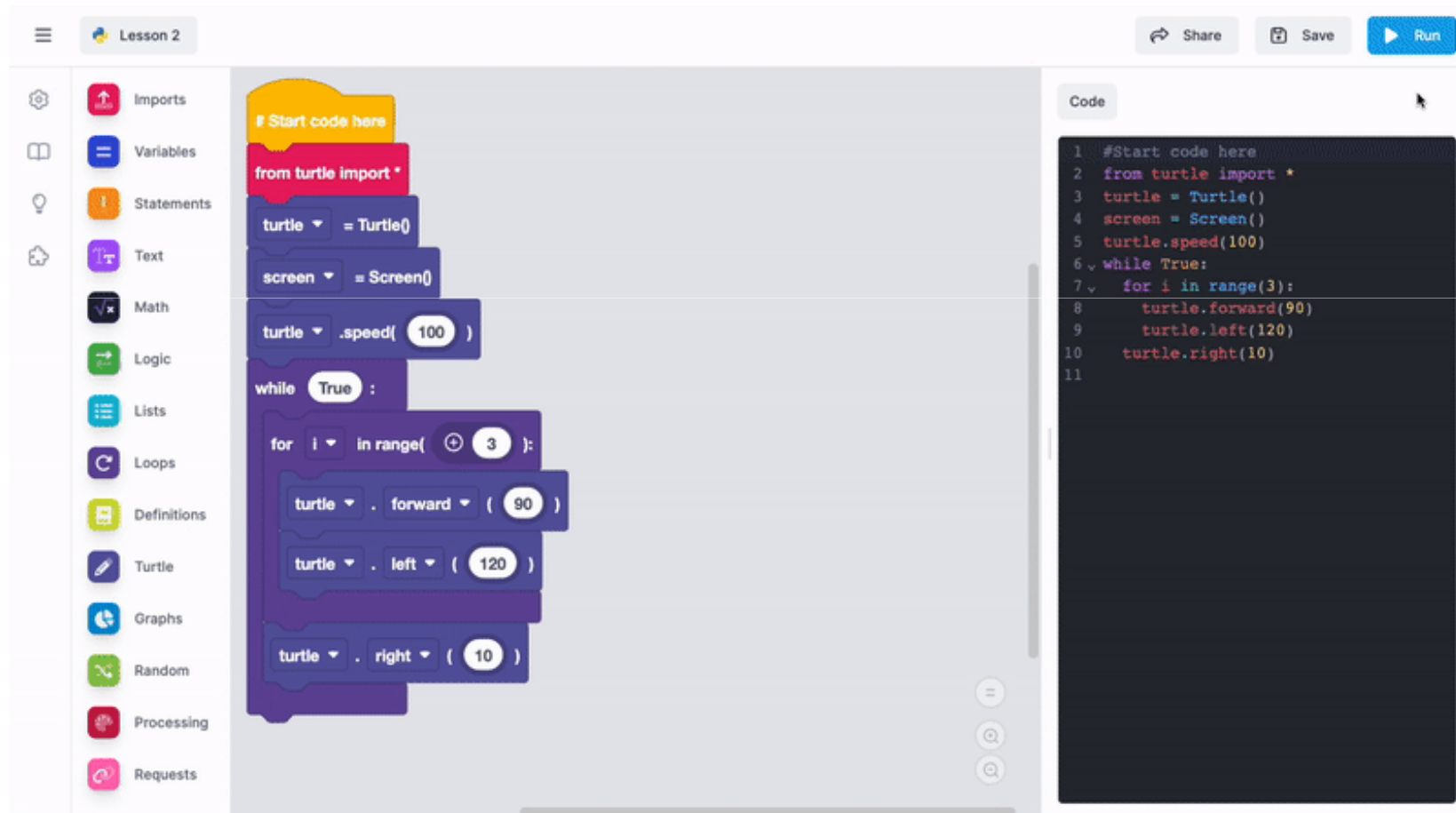
Here we modify the triangle code to draw a repeating, rotating pattern.

The **while True** loop will run forever and the **for** loop will draw the triangle

Each time the loop iterates, we move the Turtle 10 pixels

```
# Start code here
from turtle import *
turtle = Turtle()
screen = Screen()
turtle .speed( 100 )
while True :
    for i in range( 3 ):
        turtle . forward ( 90 )
        turtle . left ( 120 )
    turtle . right ( 10 )
```

Drawing patterns



The screenshot shows the Edublocks IDE interface. On the left is a sidebar with various tool categories: Imports, Variables, Statements, Text, Math, Logic, Lists, Loops, Definitions, Turtle, Graphs, Random, Processing, and Requests. The main workspace contains a Python script written in a block-based editor. The script is as follows:

```
# Start code here
from turtle import *
turtle = Turtle()
screen = Screen()
turtle.speed(100)
while True:
    for i in range(3):
        turtle.forward(90)
        turtle.left(120)
    turtle.right(10)
```

On the right side of the IDE, there is a 'Code' panel showing the same script in a text editor format. The code is:

```
1 #Start code here
2 from turtle import *
3 turtle = Turtle()
4 screen = Screen()
5 turtle.speed(100)
6 while True:
7     for i in range(3):
8         turtle.forward(90)
9         turtle.left(120)
10        turtle.right(10)
11
```

At the bottom center of the image, there is a colorful logo that reads 'PI CODE CLUB'.

Changing Colour

The colour and thickness of the “pen” can be changed

Colours are mixed using RGB values.

Values start at 0 and end at 255

Red is 255,0,0

Green is 0,255,0

Blue is 0,0,255

Thickness is called “width”

```
# Start code here
from turtle import *
turtle = Turtle()
screen = Screen()
turtle .speed( 100 )
turtle .pencolor( 255 , 0 , 0 )
turtle .width( 20 )
while True :
  for i in range( 3 ):
    turtle . forward ( 50 )
    turtle . left ( 120 )
  turtle . right ( 10 )
```

Challenge: Design a pattern

Can you create a pattern using Turtle?

- Your pattern should repeat a shape
- The pen should change colour at least once
- The width of the pen should change at least once

An example pattern

Here, two loops are used.

The first draws a red triangle at double thickness.

The second draws a blue octagon.

Did you spot the block to increase the speed of the turtle?

PI CODE CLUB

```
# Start code here
from turtle import *
turtle = Turtle()
screen = Screen()
turtle .speed( 100 )
while True :
    turtle .pencolor( 255 , 0 , 0 )
    turtle .width( 2 )
    for i in range( 3 ):
        turtle . forward ( 50 )
        turtle . left ( 120 )
    turtle . right ( 10 )
    turtle .pencolor( 0 , 0 , 255 )
    turtle .width( 2 )
    for i in range( 3 ):
        turtle . forward ( 50 )
        turtle . left ( 45 )
    turtle . right ( 10 )
```

Drawing a circle

The turtle module has a special block for drawing circles.

It uses the radius to determine the size.

Here we use a for loop to draw a circle pattern

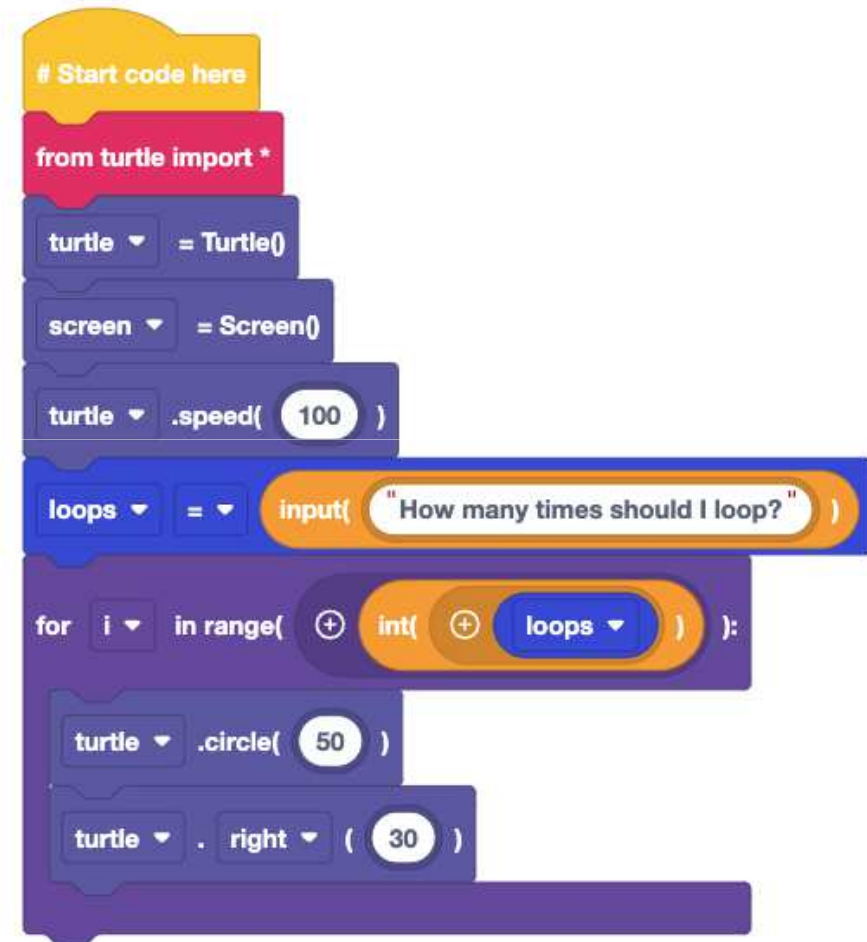
```
# Start code here
from turtle import *
turtle = Turtle()
screen = Screen()
turtle .speed( 100 )
for i in range( + 12 ):
    turtle .circle( 50 )
    turtle . right ( 30 )
```

What have we learnt?

- How to write a sequence to control Turtle
- How to use different loops for different actions
- How maths is important to create code

Next Lesson

Next Lesson we will use EduBlocks to capture user input and work with different types of data



```
# Start code here
from turtle import *
turtle = Turtle()
screen = Screen()
turtle .speed( 100 )
loops = input( "How many times should I loop?" )
for i in range( int( loops ) ):
    turtle .circle( 50 )
    turtle . right ( 30 )
```

qwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxc
vbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfgh
jklzxcvbnmqwertyuiopa
sdfghjklzxcvbnmqwerty
uiopasdfghjklzxcvbnmq
wertyuiopasdfghjklzxcv
bnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghj
klzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopa
sdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwerty
uiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrt
yuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnm
qwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxc

Edublock

Lesson 3: Lesson Plan

PI CODE CLUB

Lesson 3:

Introduction

The class will be introduced to capturing user input and using that input to control a simple application made in Turtle. They will learn about data types and what each type can be used for.

Learning Objectives

How to use edublocks

How to capture user input

How to identify different data types.

Key Vocabulary

Sequence, selection and iteration, data types (string, integer, float)

Preparation

Subject Knowledge

An understanding of creating code in a block based environment and understanding how key coding concepts are across all languages and can be illustrated using different languages.

Possible Misconceptions

User input is just keyboard input
That data types are not important.
Logical thinking is difficult.

Pedagogy

Ideally the class will each have access to a computer and complete the tasks individually. The lesson can be completed with 1 computer per 2 children.

You will need

A computer running Windows / Mac or Linux or Chromebook
A web browser (Firefox, Edge, Google Chrome, Safari)

Assessment Opportunities

Understand that coding concepts are not tied to just one language and that skills learnt in one language can be applied to others.
Understanding the logical operators (slide 8).
Understanding data types and where they are best used.

Outline Plan

This is a quick overview of the accompanying slide deck.

<p>Introduction Group Activity (Slide 3) 5 Minutes</p>	<p>Challenge the class to identify as many forms of input as possible. Including newer forms of input.</p> <p>Write them down on the board.</p> <p>For example</p> <p>Keyboard, mouse, voice, joypad, XBOX Kinect, Wiimote etc.</p> <p>In Python, user input is typically referring to keyboard input.</p>
<p>Group activity (Slide 4) 5 Minutes</p>	<p>To start Edublocks, ask the class to open a web browser and type in</p> <p>app.edublocks.org/editor</p> <p>Select Python 3 as the mode, give the project a file name and click Create</p> <p>The class are tasked with creating the code from this slide and then running the code.</p> <p>The for loop (for i in range) is deliberately going to fail. As the user input is captured as a string, which cannot be used to define the number of times that the loop iterates.</p>

<p>Group Activity 5 Minutes Slides (5 - 6)</p>	<p>Here we debug the error. This is a type error, identifying that we used an incorrect data type for the loop. But the Python error is quite scary.</p> <p>The three key data types we are focusing on are</p> <ul style="list-style-type: none"> ● String: A sequence of characters, including punctuation, numbers and letters. ● Integer: A number with no decimal place. ● Float: A number with a decimal place. <p>Challenge the class to identify the best data type for a situation.</p> <p>For example “Which data type would be useful for currencies / temperature / scientific data?” = float</p> <p>“Which data type would be useful for a message?” = string</p> <p>“Which data type would be useful to count the number of buses on a road?” = integer</p>
<p>Group Activity 5 Minutes (Slide 7)</p>	<p>Here the class will fix the issue in their code. The easiest method to fix this is to pull the input() block out, then drop the int(1) block from Statements into the now vacant space. Then drop the input() block into the int(1) block.</p>
<p>Group Activity 5 Minutes (Slide 8)</p>	<p>Here we introduce conditional statements, tests that will use logic to determine an outcome.</p> <p>In this example the class will see that if the user types in Ed, or Aijaz, then the code will greet them by name. But if they use another name, then the else condition is used, and a generic hello is printed.</p>

<p>Group Activity 5 Minutes (Slide 9)</p>	<p>Conditional statements / tests use logical operators for comparison.</p> <p>== Values must be equal.</p> <p>!= Values must NOT be equal.</p> <p>< Left value is less than the right value.</p> <p><= Left value is less than or equal to right value.</p> <p>> Left value is greater than the right value.</p> <p>>= Left value is greater than or equal to right value.</p> <p>These logical operators will provide lots of functionality for future projects.</p>
<p>Group Activity 10 Minutes (Slide 10)</p>	<p>The class should work through this example to get used to how conditional statements work and how logic is applied.</p>
<p>Group Activity 10 Minutes (Slides 11-12)</p>	<p>In this challenge the class are tasked with fixing this broken sequence of code (slide 10).</p> <p>The solution is shown on slide 12. The class is asked to make one change at a time.</p> <ul style="list-style-type: none">● What happened?● Does it work as they expected?● How can they fix it?

Plenary 2 Minutes (Slide 13)	Here we recap the learning from this lesson. <ul style="list-style-type: none">● How we captured user input.● How we modified the data type for use in specific tasks.● How we used conditional statements to change the outcome of the code.● How logical operators are used in conditional statements.
Next Time 1 Minute (Slide 14)	We will continue our learning with edublocks, and learn how to create variables that will store data, and then use the data with conditional statements.



Lesson 3

User Input & Data Types

PI CODE CLUB



By the end of this lesson, you will..

- Understand how to use EduBlocks
- Understand how to capture user input
- Understand how to identify different data types

What is Input?

How many forms of input can you name?

Traditionally user input refers to keyboard input



How can we use the input?

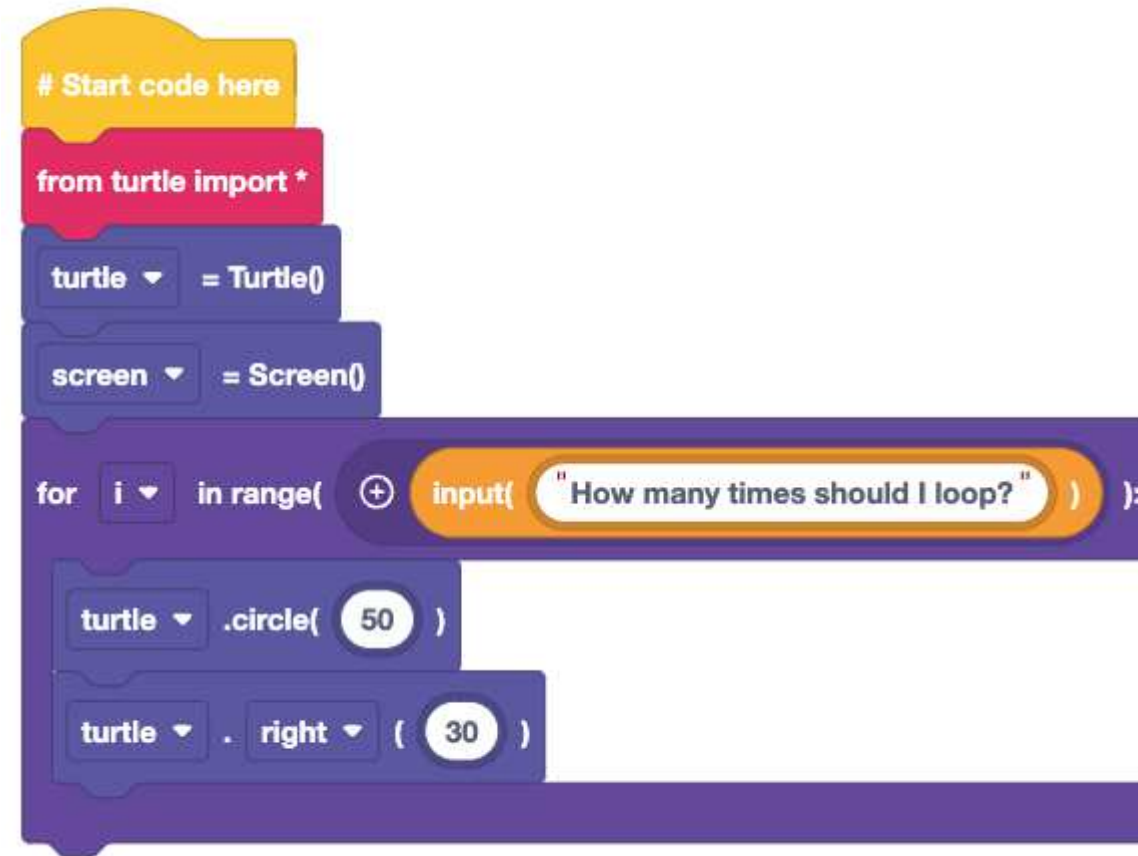
User input is a way that the computer can ask a question

The input block is found in the statements section

What will this code do?

What happens when you run it?

Why does it do that?



```
# Start code here
from turtle import *
turtle = Turtle()
screen = Screen()
for i in range( + input( "How many times should I loop?" ) ):
    turtle .circle( 50 )
    turtle .right ( 30 )
```

What happened?

There was an error!

But what does this mean?

Data types tell a computer how we want to use the data

```
TypeError: start must be a integer on line 6 in main.py
```

For a loop, we need to use an integer, a number with no decimal place

But a user input captures the number as a string, a sequence of characters

Basic data types

At the basic level there are three data types

String

A sequence of characters, including punctuation, numbers and letters.

Integer

A number with no decimal place

Float

A number with a decimal place

Give examples of where each of these data types can be used

How can we fix the error?

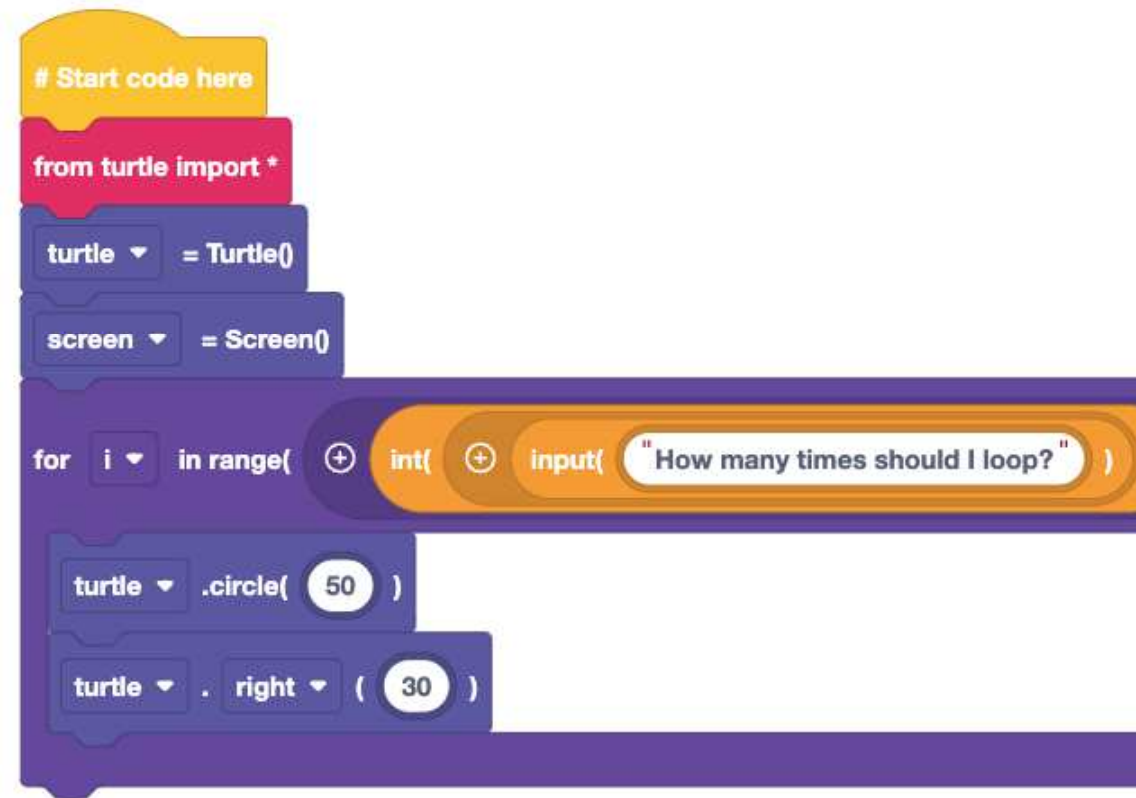
We need to convert the string to an integer

Use the int() block from statements

Wrap it around the input() block

What happens?

How can we improve the code?



```
# Start code here
from turtle import *
turtle = Turtle()
screen = Screen()
for i in range( int( input( "How many times should I loop?" ) ) ):
    turtle .circle( 50 )
    turtle .right ( 30 )
```

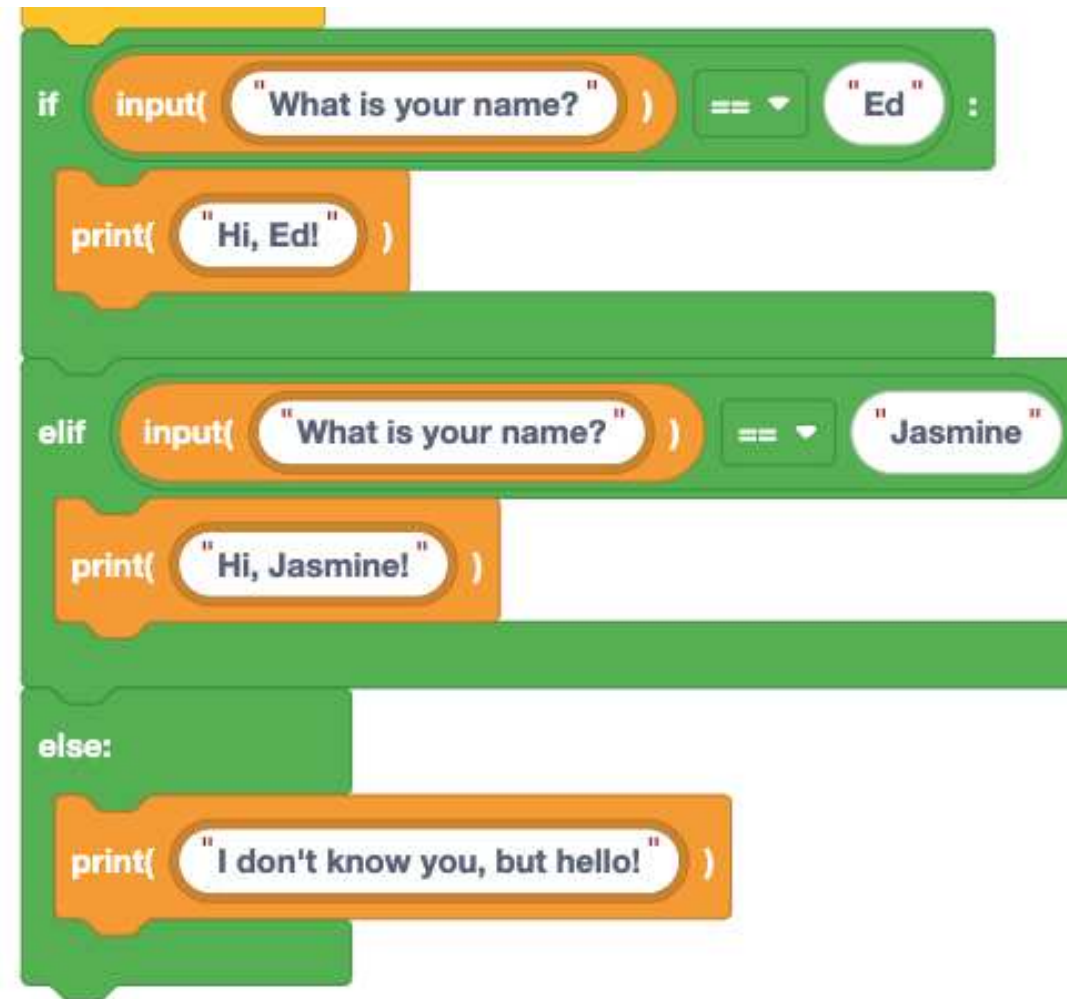
f, else if and else

How do we make a decision?

How does a computer?

Here if we answer with either "Ed" or "Jasmine" then the computer will say hello.

But if it does not know our name, it will just say hello.



```
if input( "What is your name?" ) == "Ed" :
    print( "Hi, Ed!" )
elif input( "What is your name?" ) == "Jasmine" :
    print( "Hi, Jasmine!" )
else:
    print( "I don't know you, but hello!" )
```


Logical Thinking

Our conditional test uses logic to check the user input against a value

values must be equal

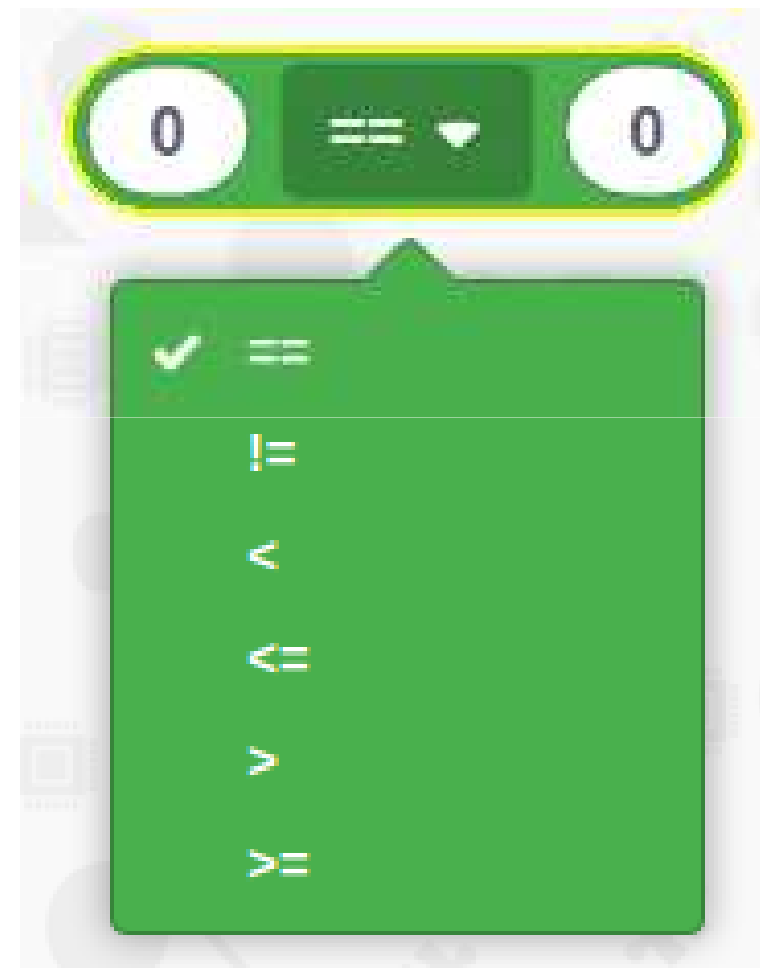
values must NOT be equal

left value is less than right value

left value is less than or equal to right value

left value is greater than right value

left value is greater than or equal to right value



Drawing shapes, logically

In this sequence of code, we use logic to draw one of two shapes. If the user input is square, then a for loop is used to draw a square. Otherwise if the user input is a circle, a circle is drawn. If we type in something else, the else condition will activate to apologise to the user.

```
# Start code here
from turtle import *
turtle = Turtle()
screen = Screen()
for i in range( int( input( "How many times should I loop? " ) ) ):
    turtle .circle( 50 )
    turtle . right ( 30 )
```

Challenge!

Can you write some code to ask...

What width should the pen be?

How many times should the code loop?

What angle should the turn be?

Where do we add/make changes to the code. Change one section at a time, run the code and see what happens!

```
# Start code here
from turtle import *
turtle = Turtle()
screen = Screen()
while True :
    turtle .width( 20 )
    for i in range( + 4 ):
        turtle .circle( 50 )
        turtle . left ( 90 )
```

Challenge Solution

Does this code look like yours?

The pen width user input was a string, as the `turtle.width()` command does not rely on a set data type

But our loop and turning angle answers require an integer to work

```
# Start code here
from turtle import *
turtle = Turtle()
screen = Screen()
while True:
    turtle.width(int(input("Pen Width?")))
    for i in range(int(input("How many loops?"))):
        turtle.circle(50)
        turtle.left(int(input("Turning angle?")))
```

What have we learnt?

- How to capture user input
- How to use specific data types for tasks
- How to use conditional logic in our code

Next Lesson

In the next lesson we will use
Scratch/duBlocks to capture user input
into a variable and use that
with conditional logic

```
# Start code here
from turtle import *
turtle = Turtle()
screen = Screen()
turtle.speed(100)
loops = input("How many times should I loop? ")
for i in range(int(loops)):
    turtle.circle(50)
    turtle.right(30)
```

Edublock

Lesson 4 – Lesson Plan

PI CODE CLUB

Lesson 4:

Introduction

The class will be introduced to variables using Edublocks by creating an application that will draw different shapes on the screen.

Learning Objectives

How to use edublocks

To understand how a sequence of code works.

How to use conditional statements to change the output of the code.

How to create and update variables.

How variables are useful tools to temporarily store data.

Key Vocabulary

Sequence, selection and iteration, variables.

Preparation

Subject Knowledge

An understanding of creating code in a block based environment and understanding how key coding concepts are across all languages and can be illustrated using different languages.

Possible Misconceptions

That variables are complicated.

Pedagogy

Ideally the class will each have access to a computer and complete the tasks individually. The lesson can be completed with 1 computer per 2 children.

You will need

A computer running Windows / Mac or Linux or Chromebook

A web browser (Firefox, Edge, Google Chrome, Safari)

Assessment Opportunities

Understand that variables are temporary storage.

How to update the contents of a variable in the sequence.

How to read the contents of a variable.

Outline Plan

This is a quick overview of the accompanying slide deck.

<p>Group Activity (Slide 3) 5 Minutes</p>	<p>Here we introduce the concept of variables using a box analogy.</p> <p>The cardboard box is a container for data, we write a name upon the box, and then place items into the box.</p> <p>To look inside the box we use the given name and Python will show us the contents.</p> <p>If you have a cardboard box to hand, along with some small items, you can demonstrate this in class.</p>
<p>Group activity (Slide 4-5) 5 Minutes</p>	<p>To start Edublocks, ask the class to open a web browser and type in</p> <p>app.edublocks.org/editor</p> <p>Select Python 3 as the mode, give the project a file name and click Create</p> <p>In these two slides we create a variable called "bedroom" and in there we shall update the contents to show three common items in a bedroom.</p> <p>The data stored in the variable is a string, can any of the class correctly identify this based on Lesson 3?</p>

<p>Group Activity 5 Minutes Slide (6)</p>	<p>The class will use this example code to learn</p> <ul style="list-style-type: none"> • How to set the contents of the loops variable to an integer. (Can they spot that?) • How to use the variable in a for loop to set the number of times that the loop will iterate. <p>Can the class identify the shape that this code will draw?</p>
<p>Group Activity 10 Minutes (Slide 7 - 8)</p>	<p>Using the code from previous slide the class will</p> <ul style="list-style-type: none"> • Add code to ask the user “How many sides the shape has?” • Use a calculation to determine the turning angle. <p>Remember that the user input will need to be saved as an integer, otherwise the code will error! The turning angle calculation is $360 / \text{number of sides}$.</p>
<p>Group Activity 10 Minutes (Slide 9 - 10 - 11)</p>	<p>We add more code to the sequence.</p> <p>A new variable “colour” is used to capture a choice of red, green or blue. Can the class remember what data type this is?</p> <p>Slide 9 introduces three conditional tests, one for each colour choice. The complete code for red is on the slide, the class is challenged to add the code for green and blue.</p> <p>Remember that the colours are set using three numbers.</p> <p>Red = 255,0,0 Green = 0,255,0 Blue = 0,0,255</p> <p>The complete solution is on slide 11</p>

<p>Plenary 2 Minutes (Slide 12)</p>	<p>Here we recap the learning from this lesson.</p> <ul style="list-style-type: none">• How to create, update and use a variable.• How we can use the data in a variable to control a loop.• How to use conditional logic with a variable
<p>Next Time 1 Minute (Slide 13)</p>	<p>We will continue our learning with edublocks, and learn how to create functions that will draw patterns and shapes automatically.</p>



Lesson 4

Variables

PI CODE CLUB



By the end of this lesson, you will...

- Understand how a sequence of code works
- Understand how to use conditional statements to change the output of the code
- Understand how to create and update variables
- Understand that variables are useful tools to temporarily store data

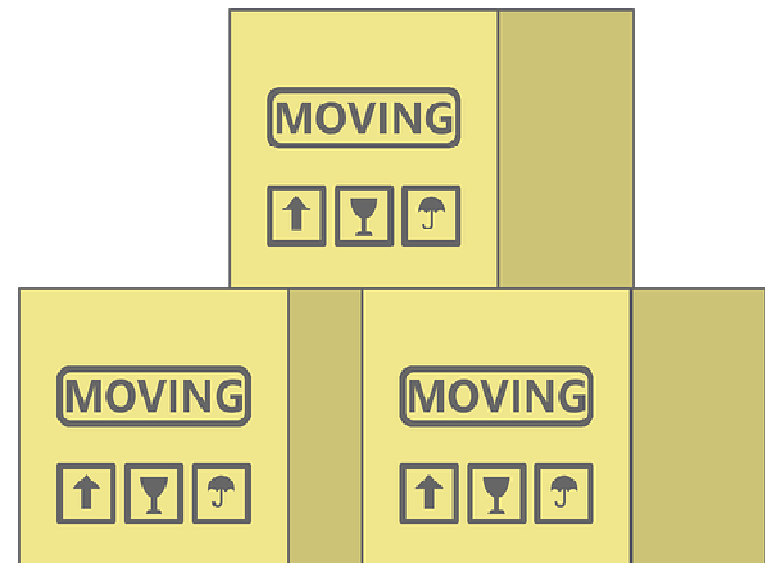
What is a variable?

Have you ever tidied your room? Or moved home?

We get a box, write a name on it, then fill it up!

The box is just like a variable, it stores things temporarily.

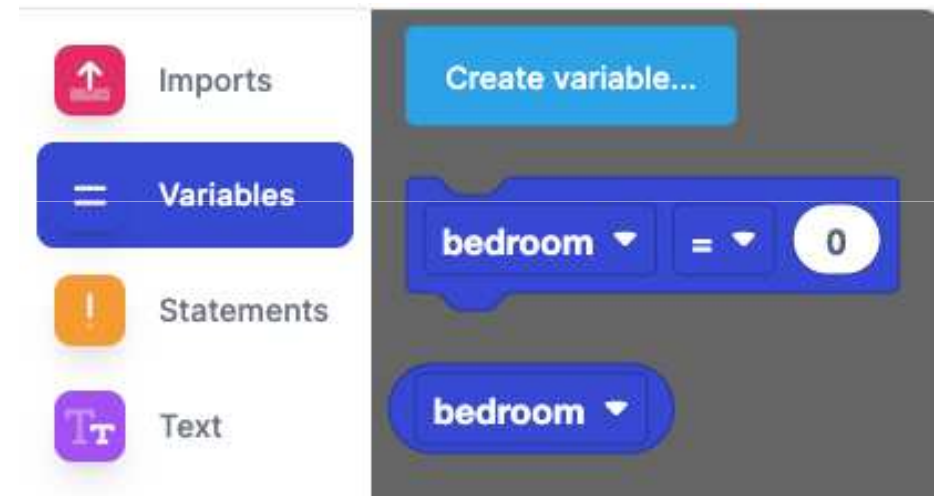
If we use the name of the box, then the contents are displayed.



How do variables work?

We first create a variable

- Go to variables
- Click on "Create Variable"
- Call the variable "bedroom"
- New blocks will appear



How do variables work?

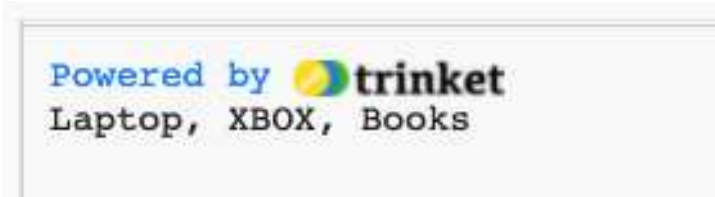
Here we fill the bedroom variable with items in our bedroom

Then we call the variable by name and print the contents

Variables can store any data type



```
# Start code here
bedroom = "Laptop, XBOX, Books"
print( bedroom )
```



```
Powered by trinket
Laptop, XBOX, Books
```


Using a variable

Create a new variable called "Loops"

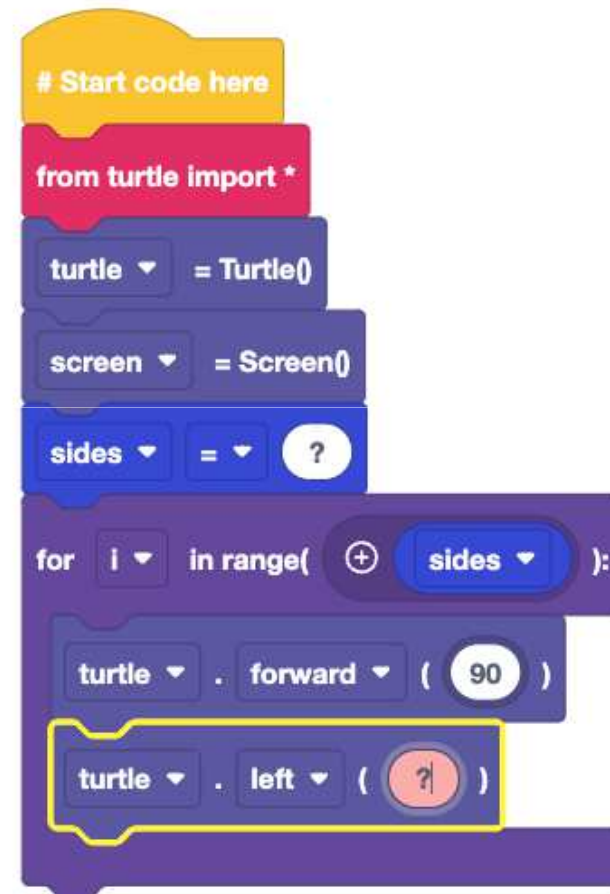
- Set the value of loops to 4
- Use a **for** loop and then get the loops block from Variables and place it in that block
- What shape does this draw?

```
# Start code here
from turtle import *
turtle = Turtle()
screen = Screen()
loops = 4
for i in range( loops ):
    turtle . forward ( 90 )
    turtle . left ( 90 )
```

Shape drawing app

We are going to create an app to draw shapes with Turtle

- How do we ask the user for the number of sides?
 - Hint: User Input
- How do we determine the turning angle?
 - Hint: The number of degrees in a circle divided by sides



```
# Start code here
from turtle import *
turtle = Turtle()
screen = Screen()
sides = ?
for i in range(+ sides):
    turtle . forward ( 90 )
    turtle . left ( ? )
```

The image shows a Scratch code editor with several blocks. At the top is a yellow 'Start code here' block. Below it is a red 'from turtle import *' block. Then two blue blocks: 'turtle = Turtle()' and 'screen = Screen()'. A blue block 'sides = ?' is highlighted with a yellow border. Below that is a purple 'for i in range(+ sides):' block. Inside the loop, there are two blue blocks: 'turtle . forward (90)' and 'turtle . left (?)'. The 'left' block is also highlighted with a yellow border.

Shape drawing app

- The sides variable stores the answer to the question "How Many Sides"
- Then we use the sides to control the number of loops
- The turning angles is the answer to the calculation $360/\text{sides}$

```
# Start code here
from turtle import *
turtle = Turtle()
screen = Screen()
sides = int(input("How many sides?"))
for i in range(sides):
    turtle.forward(90)
    turtle.left(360 / sides)
```

Adding colour to the app

- Add another variable called colour
- Store the user input with the colour variable
 - The only colours should be red, green or blue

Insert the colour variable between the sides variable and the for loop.

- Run the code
- What happens?
- Did you expect that to happen?

```
# Start code here
from turtle import *
turtle = Turtle()
screen = Screen()
sides = int( input( "How many sides?" ) )
color = input( "What color pen should I use? Red, Green or Blue?" )
for i in range( sides ):
    turtle . forward ( 90 )
    turtle . left ( 360 / sides )
```

Adding colour to the app

We have captured the users colour choice, but how do we use it?

We need to use conditional tests and logic to make this work

The green blocks are found in **Logic**

We've created the test for red, can you finish the code?

Red: 255,0,0

Green: 0,255,0

Blue: 0,0,255

Run the code, what happens?

```
# Start code here
from turtle import *
turtle = Turtle()
screen = Screen()
sides = int(input("How many sides? "))
color = input("What color pen should I use? Red, Green or Blue? ")

if color == "red":
    turtle.pencolor(255, 0, 0)
elif True:
    # Empty block
elif True:
    # Empty block

for i in range(sides):
    turtle.forward(90)
    turtle.left(360 / sides)
```

Challenge Solution

Does your code look like this?

- We've used variables to capture the user information
- Conditional logic to compare the input with known colours
- Drawn shapes using maths and loops

```
# Start code here
from turtle import *
turtle = Turtle()
screen = Screen()
sides = int(input("How many sides? "))
color = input("What color pen should I use? Red, Green or Blue? ")

if color == "red":
    turtle.pencolor(255, 0, 0)
elif color == "green":
    turtle.pencolor(0, 255, 0)
elif color == "blue":
    turtle.pencolor(0, 0, 255)

for i in range(sides):
    turtle.forward(90)
    turtle.left(360 / sides)
```

What have we learnt?

- How to create a variable
- How to use a variable to control a loop
- How to use conditional logic with a variable

Next Lesson

Next Lesson we will use EduBlocks to create functions that will draw patterns and shapes automatically

```
# Start code here
from turtle import *
turtle = Turtle()
screen = Screen()

def sides ( n ):
    for i in range( n ):
        turtle . forward ( 30 )
        turtle . left ( 360 / n )

while True :
    sides ( int( input( "How many sides does the shape have?" ) ) )
```


Edublock

Lesson 5 – Lesson Plan

PI CODE CLUB

Lesson 5:

Introduction

The class will be introduced to functions using Edublocks by creating an application that will draw different shapes on the screen, while using only one module of code.

Learning Objectives

How to use edublocks

To understand how a sequence of code works.

To understand what a function is.

To understand what a function with argument is.

How to reuse code

Key Vocabulary

Sequence, selection and iteration, functions, subroutines.

Preparation

Subject Knowledge

An understanding of creating code in a block based environment and understanding how key coding concepts are across all languages and can be illustrated using different languages.

Possible Misconceptions

That variables are complicated.

Pedagogy

Ideally the class will each have access to a computer and complete the tasks individually. The lesson can be completed with 1 computer per 2 children.

You will need

A computer running Windows / Mac or Linux or Chromebook

A web browser (Firefox, Edge, Google Chrome, Safari)

Assessment Opportunities

Understand that variables are temporary storage.

How to update the contents of a variable in the sequence.

How to read the contents of a variable.

Outline Plan

This is a quick overview of the accompanying slide deck.

<p>Group Activity (Slide 3) 5 Minutes</p>	<p>Here we introduce the concept of functions using a cup of tea.</p> <p>The goal is for children to understand that by giving the command “Make a cup of tea” a sequence of code is executed to complete the task.</p>
<p>Group activity (Slide 4-5) 10 Minutes</p>	<p>To start Edublocks, ask the class to open a web browser and type in</p> <p>app.edublocks.org/editor</p> <p>Select Python 3 as the mode, give the project a file name and click Create</p> <p>On slide 4 we start by defining a function, called circles. This function is the most basic version of a function.</p> <p>On slide 5 we develop the code which will go inside the function, this code will draw 10 circles on the screen in a rotating pattern.</p> <p>The last block in the sequence calls the circles function</p>

<p>Group Activity 5 Minutes Slide (6)</p>	<p>A function with an argument is a way for us to pass extra information to the function when we call it.</p> <p>Going back to the cup of tea analogy, we can say “Make me 2 cups of tea” with the number of cups being the argument.</p> <p>Here we create a function that will draw shapes with n number of sides.</p>
<p>Group Activity 10 Minutes (Slide 7 - 8)</p>	<p>Slide 7 highlights the blocks needed to draw a shape with any number of sides. It reuses a lot of code from previous sessions.</p> <p>Slide 8 concentrates on calling the function, and using the input block to capture the number of sides from the user.</p>
<p>Group Activity 10 Minutes (Slide 9 - 10)</p>	<p>Slide 9 covers the addition of the Turtle blocks, something that the class have done since Lesson 2. It also shows the complete code for this project.</p> <p>Slide 10 elaborates on calling the function and showing the sequence which the function handles.</p>
<p>To the class 5 Minutes (Slide 11)</p>	<p>We discuss that functions are powerful tools, which can reduce the amount of code needed in a sequence.</p> <ul style="list-style-type: none"> ● This means that we do not have to write as much code. ● We can reuse code. ● Our code is smaller and easier to understand.
<p>Plenary 2 Minutes (Slide 12)</p>	<p>Here we recap the learning from this lesson.</p> <ul style="list-style-type: none"> ● How to create a function. ● How to create a function with an argument. ● How functions can be reused for different outputs.
<p>Next Time 1 Minute (Slide 13)</p>	<p>Next lesson, we will build a project to create patterns using shapes and colours!</p>



Lesson 5

Functions

PI CODE CLUB



By the end of this lesson, you will...

- Understand how a sequence of code works
- Understand what a function is
- Understand what a function with an argument is
- Understand how to reuse code

What is a function?

“Make me a cup of tea please”

How do you make a cup of tea?

What steps are involved?

How do we know this?

A function is a command which contains the steps needed to perform a task



How does a function work?

We are going to create a function to draw 10 circles in a pattern

- Go to Definitions and drag def block into the coding area
- In the first blank, type **circles** this will define the name of the function

We have now created a function. But we need to add code to the function. This is the code that will be ran when the function is called.



Using the function

We import turtle as per previous lessons.

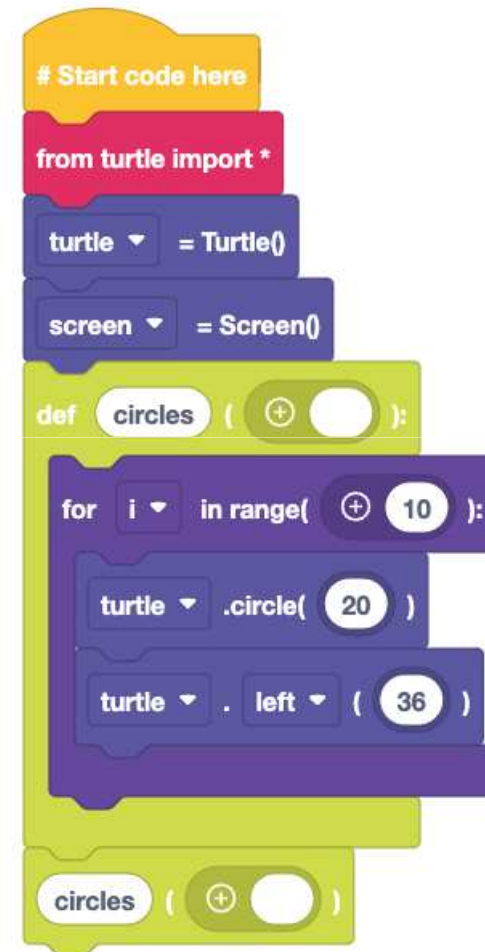
The circles function will use a for loop. It will go round 10 times.

It will draw a circle with a radius of 20 pixels, then turn left by 36 degrees (360/10)

We then call the circles function and watch as it automatically draws the pattern.

Run the code, what happens? Does it work as expected?

Can you change it to do something else?



```
# Start code here
from turtle import *
turtle = Turtle()
screen = Screen()

def circles ( ):
    for i in range( 10 ):
        turtle .circle( 20 )
        turtle . left ( 36 )

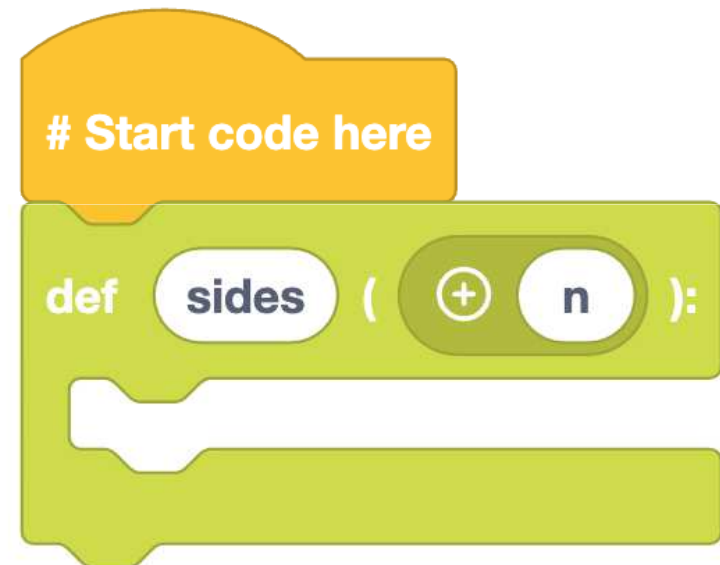
circles ( )
```

A function with an argument?

We are going to create a function to draw any shape using Turtle

- Go to Definitions and drag def block to the coding area
- In the first blank, type **sides** and in the second type n
- The name of the function is **sides** and it has an extra parameter of n. This is called an **argument**.

We have now created a function with an argument. The code inside is run when its name is used.



Code inside the function

Inside of the function we add:

- A for loop which uses n to set the range
- We also use two turtle blocks to move forward and left
 - **Can you remember what this calculation will do?**

```
# Start code here  
def sides ( + n ):  
  for i in range( + n ):  
    turtle . forward ( 30 )  
    turtle . left ( 360 / n )
```

The image shows a Scratch code editor snippet. It starts with an orange comment block "# Start code here". Below it is a green function block "def sides (+ n):". Inside the function, there is a purple "for" loop block "for i in range(+ n):". Inside the loop, there are two blue turtle blocks: "turtle . forward (30)" and "turtle . left (360 / n)".

Using a function

Inside a while True loop:

- Drag **function_name** from Definitions
- Type **sides** in the first blank
- Into the second blank, drag **int(1)** from statements
- Then drag **input()** from Statements and place on top of the **int()** block
- For the input block, ask the user “How many sides does this shape have?”



```
# Start code here
def sides ( n ):
  for i in range( n ):
    turtle . forward ( 30 )
    turtle . left ( 360 / n )
while True :
```

The image shows a Scratch code editor with the following blocks: an orange 'Start code here' block, a green 'def sides (n):' block, a purple 'for i in range(n):' block containing a 'turtle forward (30)' block and a 'turtle left (360 / n)' block, and a purple 'while True :' block with an empty input field.


Importing the turtle blocks

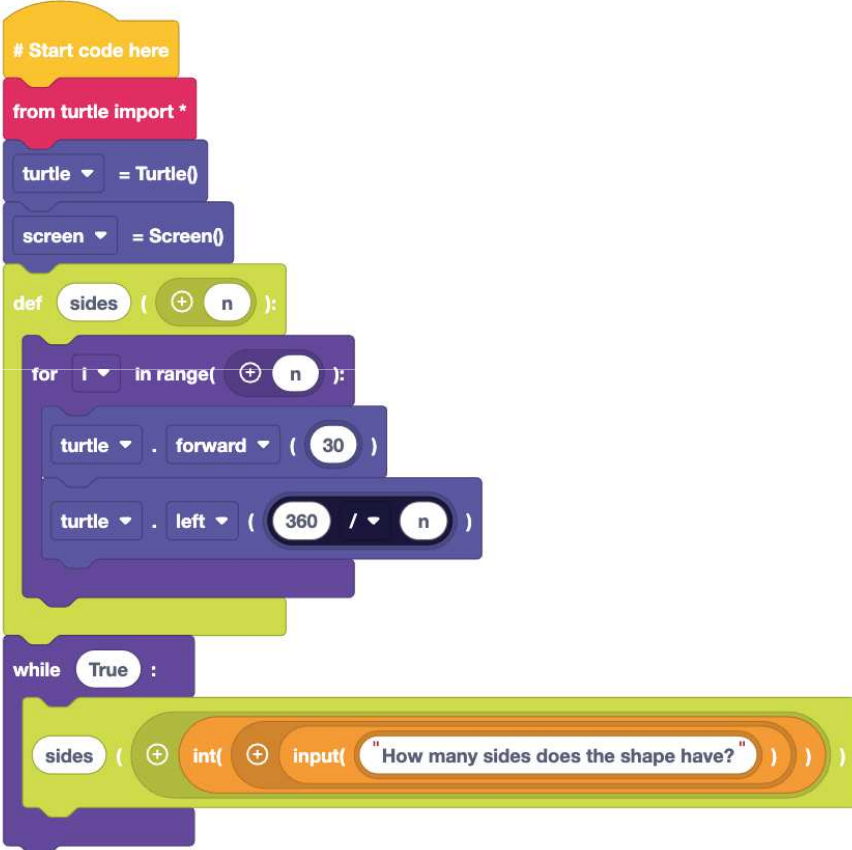
From the **import** and **turtle** blocks, we need to drag:

```
from turtle import *
```

```
Turtle = Turtle()
```

```
screen = Screen()
```

Your code should look like this,
click  to test



```
# Start code here
from turtle import *
turtle = Turtle()
screen = Screen()

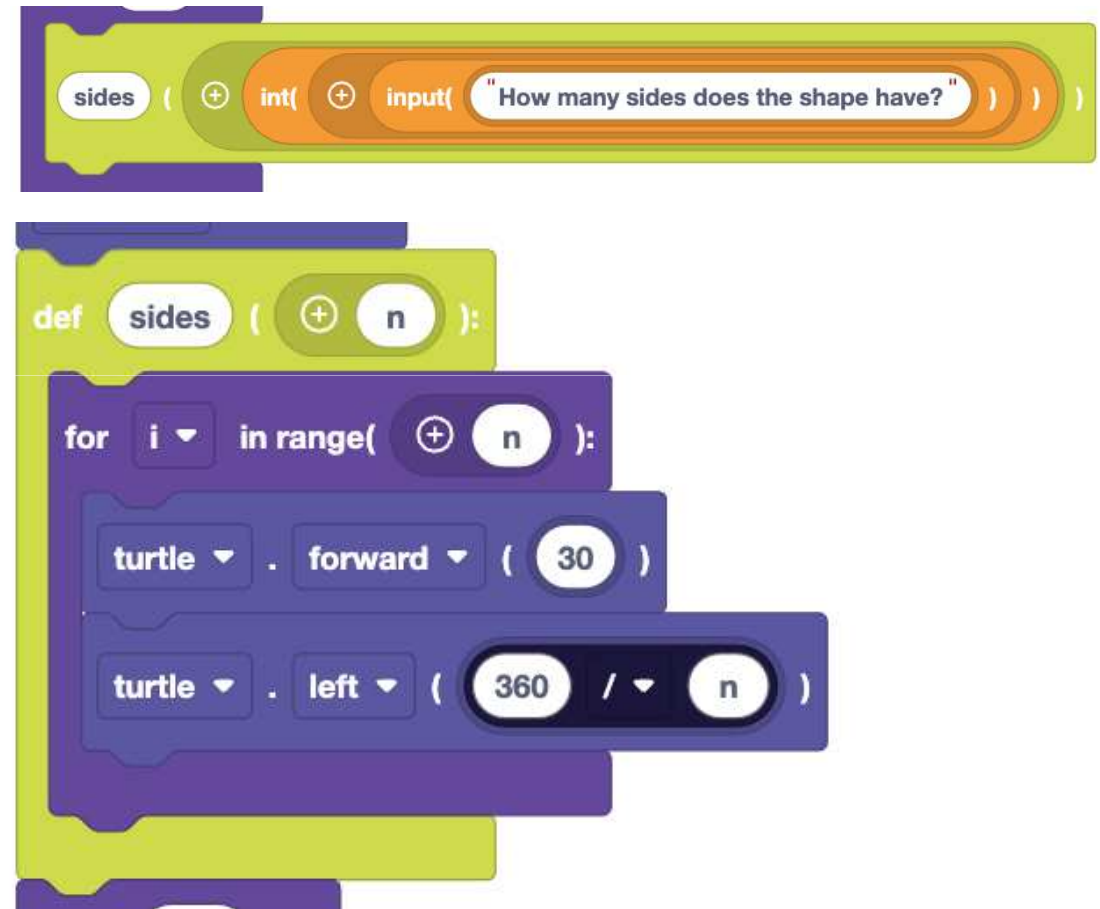
def sides ( n ):
    for i in range( n ):
        turtle . forward ( 30 )
        turtle . left ( 360 / n )

while True :
    sides ( int( input( "How many sides does the shape have?" ) ) )
```

What happens?

By using the name **sides**, we call the function.

The user input captures the number of sides the shape has. This is saved to **n** which is then used to control the for loop. It also forms part of the calculation to determine the turning angle.



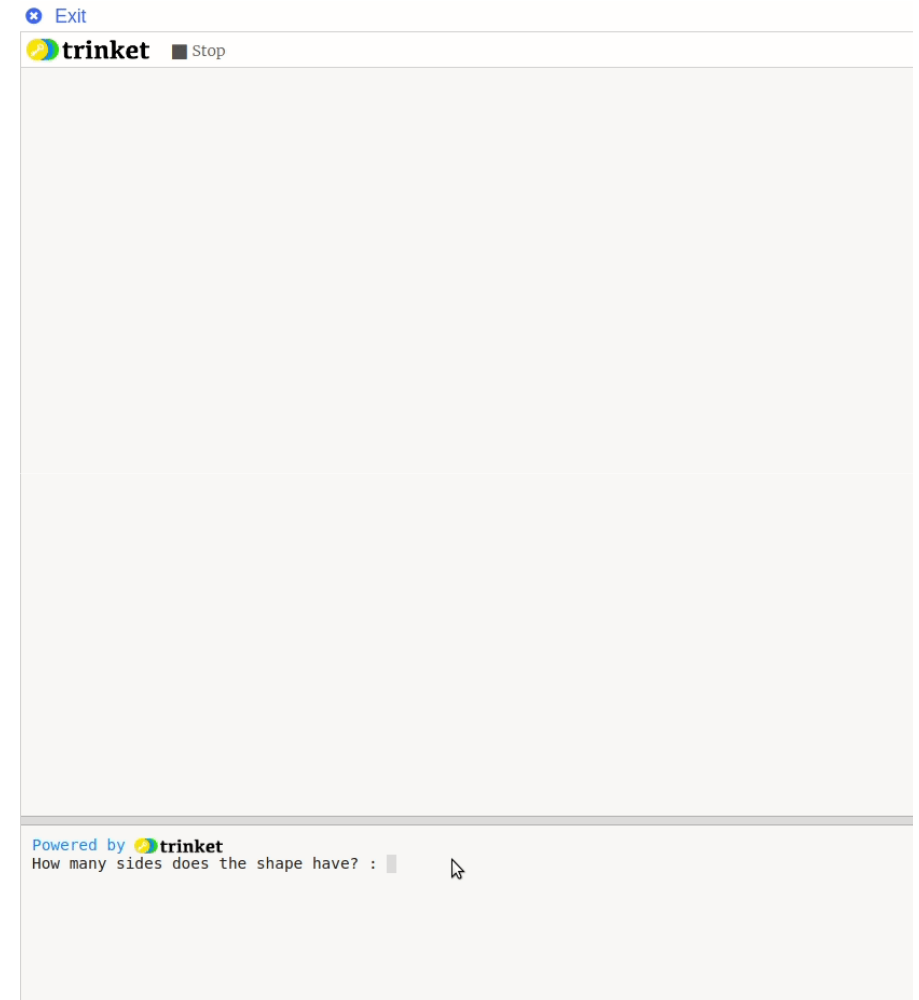
```
sides ( + int( + input( "How many sides does the shape have?" ) ) )  
  
def sides ( + n ):  
  for i in range( + n ):  
    turtle . forward ( 30 )  
    turtle . left ( 360 / n )
```

Why are functions useful?

Functions are powerful tools. They are subroutines, small sequences of code inside the main code.

We can call the function, and come out of the main code, do the function, then come back to the code.

They enable us to reuse sections of code. They keep our code tidy, and with fewer lines to write. In our code we can draw any shape using one section of code.

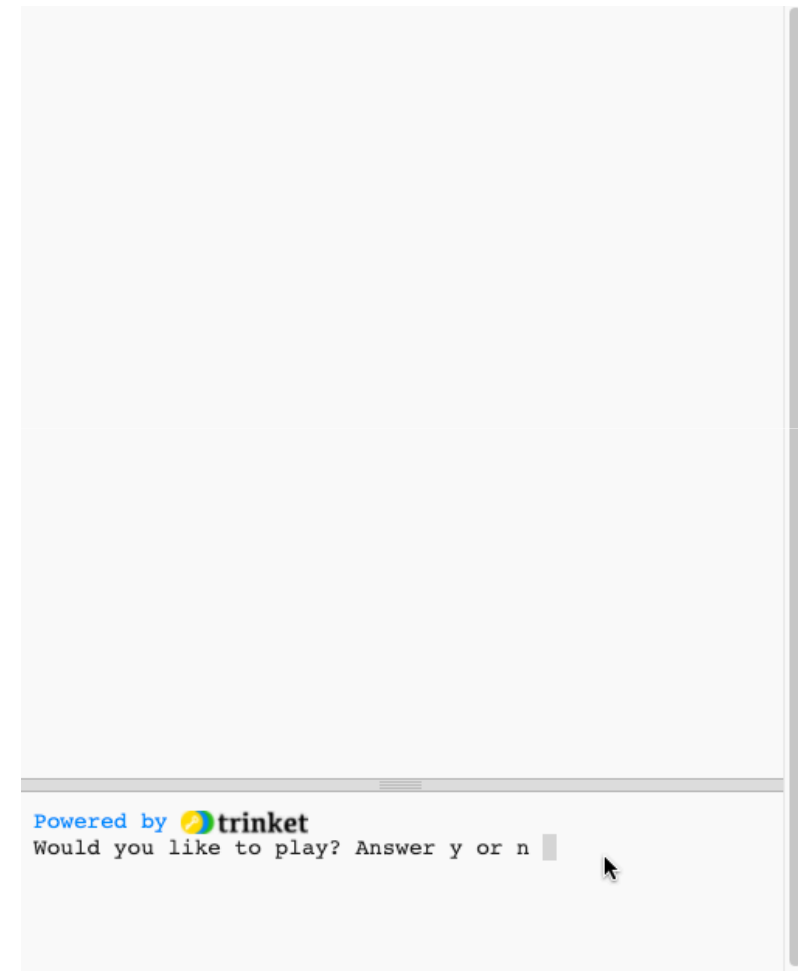


What have we learnt?

- How to create a basic function
- How to create a function with an argument
- How functions can be reused for different outputs

Next Lesson

Next Lesson we will use EduBlocks to build our own projects to create patterns using shapes and colours



PI CODE CLUB

Edublock

Lesson 6 – Lesson Plan

PI CODE CLUB

Lesson 6:

Introduction

The class will be asked to create a project using their knowledge from the past five lessons.

Learning Objectives

How to use edublocks

To understand how a sequence of code works.

To understand what a function is.

To understand what a function with argument is.

How to create and update variables.

How to create loops with definite ranges.

How to capture user input.

How to ensure the right data types are used.

How to reuse code

Key Vocabulary

Sequence, selection and iteration, functions, subroutines, loops, variables, data types

Preparation

Subject Knowledge

An understanding of creating code in a block based environment and understanding how key coding concepts are across all languages and can be illustrated using different languages.

Possible Misconceptions

That code has no creative purpose, and that code cannot be reused.

Pedagogy

Ideally the class will each have access to a computer and complete the tasks individually. The lesson can be completed with 1 computer per 2 children.

You will need

A computer running Windows / Mac or Linux or Chromebook

A web browser (Firefox, Edge, Google Chrome, Safari)

Assessment Opportunities

Understand the coding concepts that have been covered in the previous five lessons.

Creativity and free

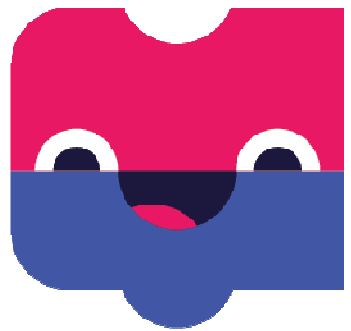
Outline Plan

This is a quick overview of the accompanying slide deck.

<p>Group Activity (Slide 3 - 4) 5 Minutes</p>	<p>The goal of this slide is to remind the class on the coding concepts that we have learnt</p> <ul style="list-style-type: none">● Sequence, the blocks / lines of code that make up a project.● Selection, using logic to determine the correct path based on input / data.● Definite iteration, count controlled loops such as for loops.● Indefinite iteration for example while True.● Subroutines are called functions in Python and they enable blocks of code to be reused without retyping their contents.● Assignment, where we save and use data stored in variables.
---	---

<p>Group activity (Slide 5) 5 Minutes</p>	<p>To start Edublocks, ask the class to open a web browser and type in</p> <p>app.edublocks.org/editor</p> <p>Select Python 3 as the mode, give the project a file name and click Create</p> <p>The class should already be familiar with these blocks, they have been using them for the past four lessons.</p> <p>Two new blocks are</p> <ul style="list-style-type: none"> • <code>screen.bgcolor(0,0,0)</code> <ul style="list-style-type: none"> ○ This sets the background colour to black. • <code>turtle.speed(100)</code> <ul style="list-style-type: none"> ○ Increases the speed at which the turtle moves around the screen.
<p>Group Activity 5 Minutes Slide (6)</p>	<p>This code continues from the previous slide.</p> <p>Here we introduce a question to the user, do they want to play the game? If they answer y, then the code inside the if condition activates. Anything else will end the code.</p>
<p>Group Activity 10 Minutes (Slide 7 - 8)</p>	<p>This code continues from the previous slide.</p> <p>Slide 7 shows the code to draw a star on the screen.</p> <p>Slide 8 will draw a circle, using the radius of the circle to set the size</p>
<p>Group Activity 5 Minutes (Slide 9)</p>	<p>This code continues from the previous slide.</p> <p>We ask the user to set the radius of the circle by capturing the user input. This is then saved as an integer to a variable called number.</p> <p>Can the class remember where they make variables?</p>

<p>Group Activity 5 Minutes (Slide 10)</p>	<p>This code continues from the previous slide.</p> <p>Using a for loop means we can call the star() function to draw a star, then change the pen colour, then draw another star.</p>
<p>Group Activity 5 minutes (Slide 11)</p>	<p>This code continues from the previous slide.</p> <p>Here a for loop (definite iteration) is used to draw eight green circles on the screen.</p>
<p>Group Activity 10 minutes (Slide 12)</p>	<p>All of the code for this project is available as a download</p> <p>Project 6 Example - The complete code listing.</p> <p>This code will need to be opened in Edublocks via the open menu. You can then scroll through the code with the class, and show it working.</p> <p>Lesson 6 Extension.xml - Extension activity, all of the alterations made.</p> <p>This is the code that shows off the extra features requested.</p>
<p>Plenary 2 Minutes (Slide 13)</p>	<p>Here we recap what we have learnt in this lesson.</p>
<p>Congratulations 1 Minute (Slide 14)</p>	<p>All of the lessons have now been completed. Well done!</p>



Lesson 6

Project

PI CODE CLUB



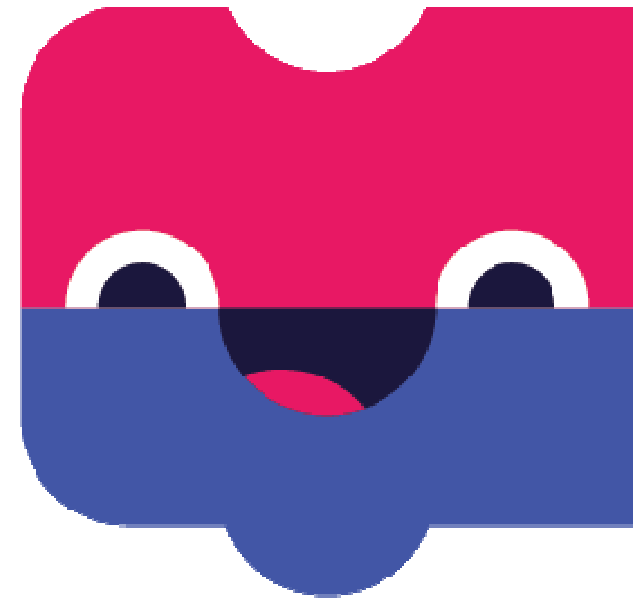
By the end of this lesson, you will...

- Understand how a sequence of code works
- Understand what a function is
- Understand what a function with an argument is
- Understand how to create and update variables
- Understand how to create loops with definite ranges
- Understand how to capture user input
- Understand how to ensure the right data types are used
- Understand how to reuse code

Creating a project

In this lesson, we shall use all of our knowledge to create a custom pattern generator using EduBlocks and Turtle

Can you remember all of the coding concepts that we learnt?



PI CODE CLUB

Coding Concepts

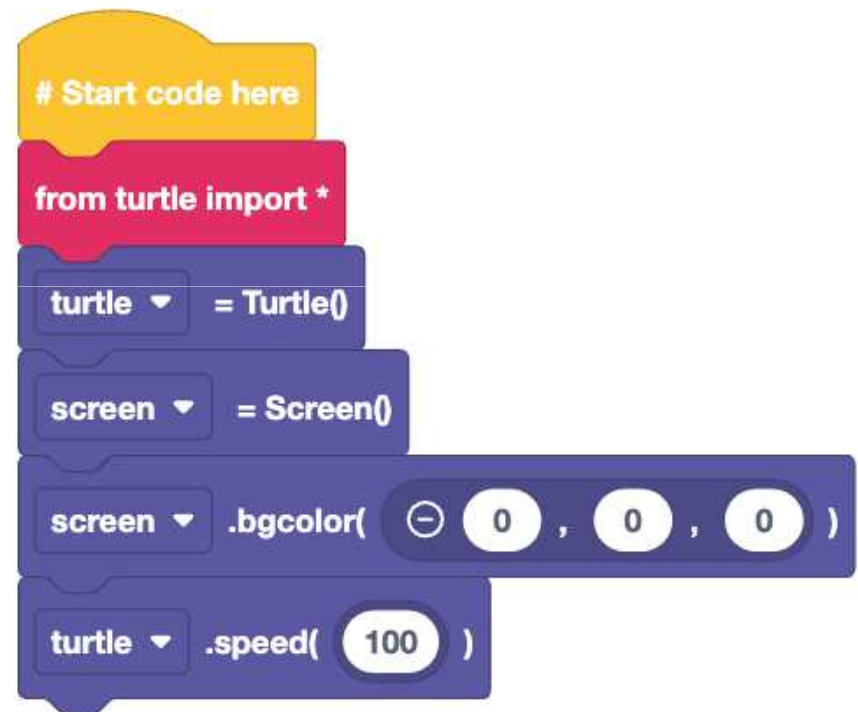
Over the past five lessons we have learnt:

- How to write a sequence
- How to use loops
- How to capture user input
- How to work with data types
- How to store data in a variable
- How to make decisions using conditional statements
- How to use functions to reuse code

Starting the project

Before we start the project we need to:

- Import the turtle module
- Connect to Turtle
- Create a screen
- Set the screen colour to Black
- Speed up the Turtle to 100!



```
# Start code here
from turtle import *
turtle = Turtle()
screen = Screen()
screen .bgcolor( 0 , 0 , 0 )
turtle .speed( 100 )
```

The image shows a stack of Scratch code blocks. At the top is a yellow comment block with the text "# Start code here". Below it is a pink block for importing the turtle module: "from turtle import *". This is followed by three dark blue blocks: "turtle = Turtle()", "screen = Screen()", and "screen .bgcolor(0 , 0 , 0)". The final block is another dark blue block: "turtle .speed(100)".

Starting the game

- Would you like to play a game?
- This is the question asked to the user before the sequence starts
- If they answer "y" the code will run
- But if they answer "n" or anything else, the else condition is activated and the game ends



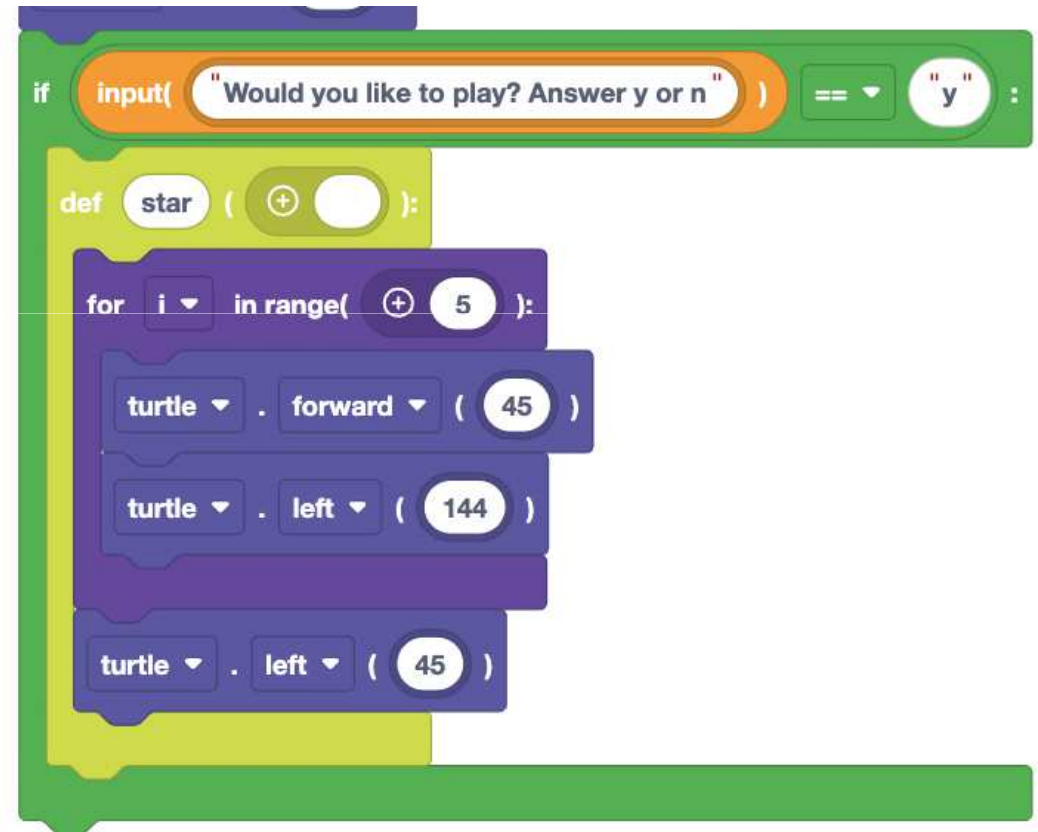
```
if input( "Would you like to play? Answer y or n" ) == "y" :  
    
else:  
  print( "Ok bye!" )
```

The image shows a Scratch code block for an if-else statement. The 'if' block contains an 'input' block with the text "Would you like to play? Answer y or n" and a comparison operator '==' followed by a 'y' block. The 'else' block contains a 'print' block with the text "Ok bye!".

What happens if...?

If the player chooses to play, then any code inside the if condition is activated.

Inside the if condition we create a function to draw a star pattern.



```
if input( "Would you like to play? Answer y or n" ) == "y" :
  def star ( + ):
    for i in range( + 5 ):
      turtle . forward ( 45 )
      turtle . left ( 144 )
    turtle . left ( 45 )
```

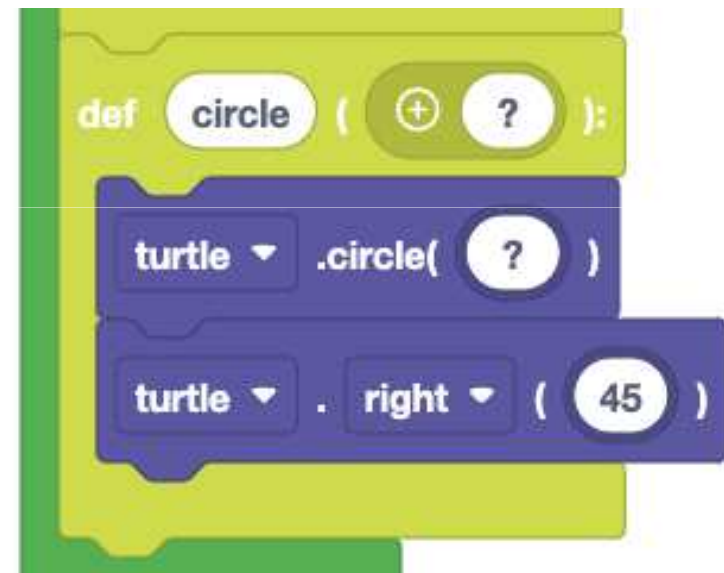
The image shows a Scratch code editor snippet. It starts with an 'if' block that checks if the user input is 'y'. Inside the 'if' block, there is a 'def' block for a function named 'star'. The 'star' function has a parameter 'i' and a range of 5. Inside the 'for' loop, there are three 'turtle' blocks: 'forward (45)', 'left (144)', and 'left (45)'. The 'def' block is highlighted in yellow, and the 'for' loop is highlighted in purple.

Draw a circle

Here we create a function to draw a circle

But, what do we replace “?” with?

Is it the diameter of the circle?
The radius?
Or the circumference?

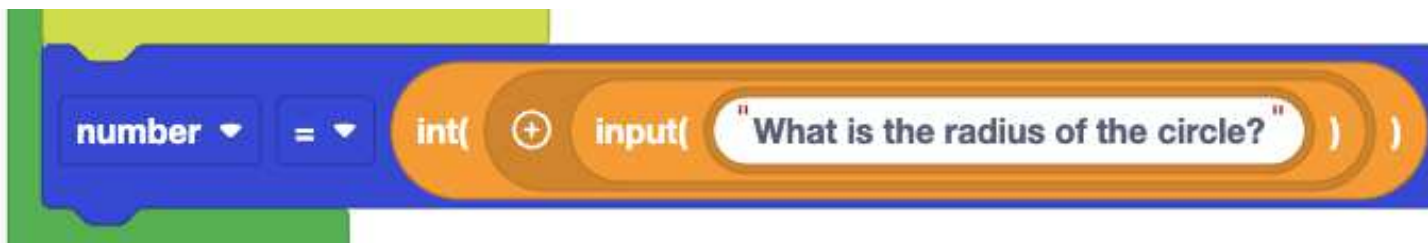


What is the radius of a circle?

To set the radius of the circle...

- We need to ask the user, via user input
- Then ensure the value given is an integer
- Save the answer to a variable called "number"

Why do we want the value to be an integer?



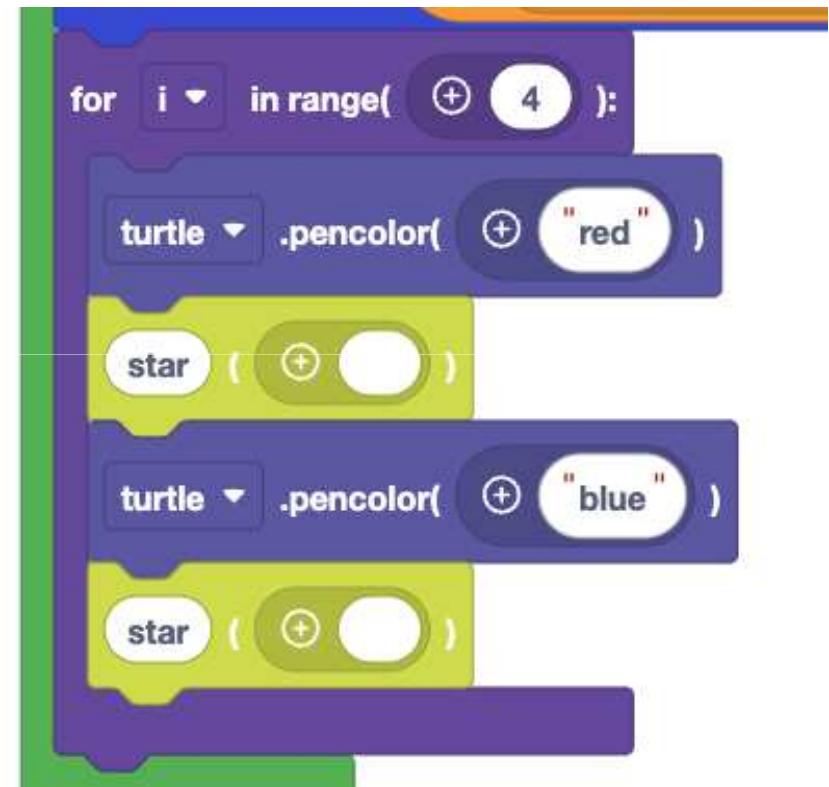
PI CODE CLUB

Draw 4 stars

Using a loop is a smart move

We can write code once, and run it many times

Here we set the Turtle pen colour to "red" and then "blue, in between we call the star function



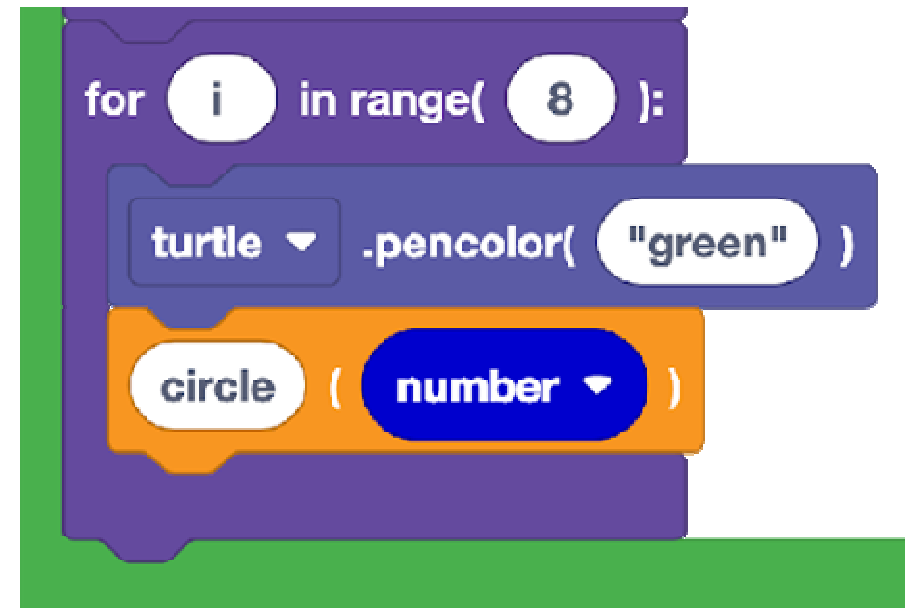
```
for i in range(4):
  turtle.pencolor("red")
  star()
  turtle.pencolor("blue")
  star()
```

The image shows a Scratch code editor snippet. It features a purple 'for' loop block with 'i' as the loop variable and '4' as the range. Inside the loop, there are four blocks: a purple 'turtle.pencolor("red")' block, a green 'star()' block, a purple 'turtle.pencolor("blue")' block, and another green 'star()' block. The blocks are connected by a vertical line on the left side of the loop block.

8 Circles

To draw 8 circles we use another for loop

We set the pen colour to green, and then call the circle function, using the data saved in the number variable



Complete Code Listing

Here is all of the code in EduBlocks.
Your teacher will now open the
code and can zoom in for clarity.

Can you alter the code to

- Draw another shape?
- Ask the user how many circles to draw?
- Put all of the code in a while True loop

Can you also add new features?

PI CODE CLUB

```
3 Start code here
from turtle import *
turtle = Turtle()
screen = Screen()
screen = bgcolor(0, 0, 0)
turtle = speed(100)

# input: "Would you like to play? Answer y or n"
star = True
for i in range(5):
    turtle = forward(45)
    turtle = left(144)
    turtle = left(45)
circle = True
turtle = circle(7)
turtle = right(45)

number = int(input("What is the radius of the circle? "))
for i in range(4):
    turtle = pencolor("red")
    star = True
    turtle = pencolor("blue")
    star = True
    for i in range(8):
        turtle = pencolor("green")
        circle = number
else:
    print("Ok bye!")
```

What have we learnt?

- How to create a full project using code from previous lessons
- That code can be used for creative projects
- That Sequence, Selection, Iteration, Assignment and Loops are key coding concepts

Congratulations!

You have successfully completed six lessons with EduBlocks, learn how to write Python code with blocks and be creative!

Next lesson, we'll test our knowledge with an end of unit assessment



123 Coding Cards

Python 3

Beginners Level

**Updated for Scratch 3
and the new EduBlocks**

edublocks.org

Before you get started with these cards, you need to make sure you're all setup.

You will need:



PC, Mac or iPad



Internet Connection



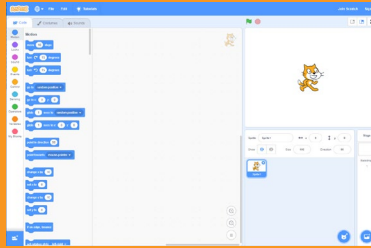
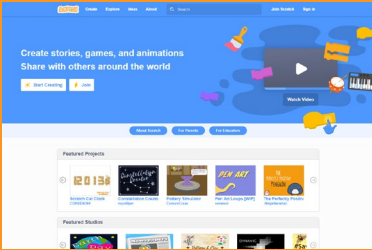
Web Browser

Scratch Setup

Go to scratch.mit.edu

Click on Create

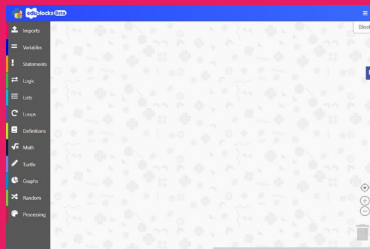
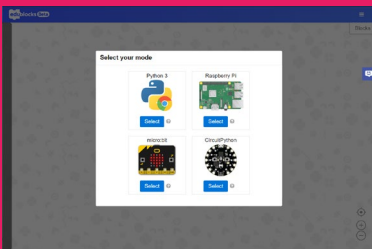
You're all setup!



EduBlocks Setup

Go to app.edublocks.org Select Python

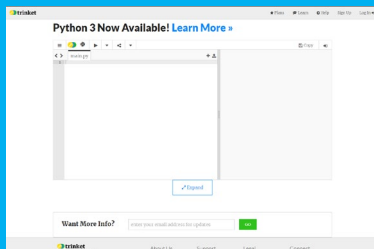
You're all setup!



Python Setup

Go to trinket.io/python3

You're all setup!



123 Coding Cards

1: Say Hello!

Scratch

say Hello! for 2 seconds



EduBlocks

```
import time
```

```
print(" Hello! ")
```

```
time.sleep( 2 )
```

Hello!

Python

```
import time  
print("Hello!")  
time.sleep(2)
```

Hello!

How to run your code:

Scratch



Click the first block

EduBlocks



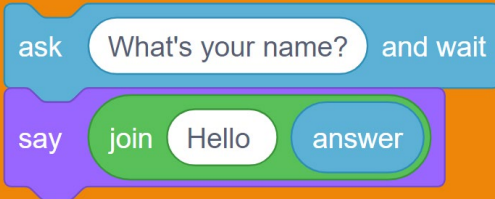
Click the Run Button

Python

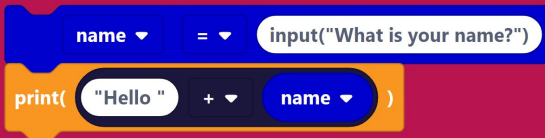


Press Play

Scratch



EduBlocks



What is your name? George
Hello George

Python

```
name = input("What is your name?")  
print("Hello " + name)
```

What is your name? George
Hello George

How to run your code:

Scratch



Click the first block

EduBlocks



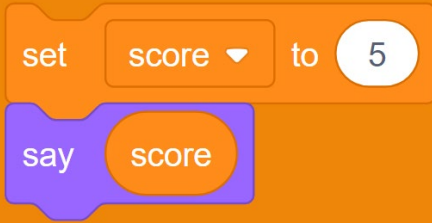
Click the Run Button

Python

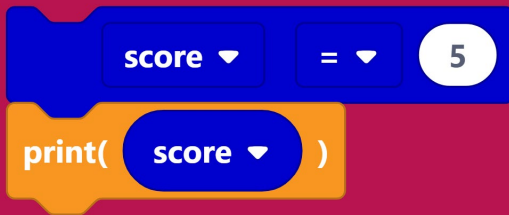


Press Play

1 Scratch



2 EduBlocks



5

3 Python

```
score = 5  
print(score)
```

5

How to run your code:

Scratch



Click the first block

EduBlocks



Click the Run Button

Python



Press Play

Scratch

```
set character to length of Hello!  
say character
```



EduBlocks

```
character = len("Hello!")  
print(character)
```

6

Python

```
character = len("Hello!")  
print(character)
```

6

How to run your code:

Scratch



Click the first block

EduBlocks



Click the Run Button

Python



Press Play

Scratch

```
forever
say Hello World!
wait 1 seconds
```



EduBlocks

```
import time
while True :
print(" Hello World! ")
time.sleep( 1 )
```

Hello World!
Hello World!

...

Python

```
import time
while True:
print("Hello World!")
time.sleep(1)
```

Hello World!
Hello World!

...

How to run your code:

Scratch



Click the first block

EduBlocks



Click the Run Button

Python



Press Play

Scratch

say

pick random

1

to

6



EduBlocks

```
import random
```

```
print( random.randint( 1,6 ) )
```

2

Python

```
import random  
print(random.randint(1,6))
```

3

How to run your code:

Scratch



Click the first block

EduBlocks



Click the Run Button

Python



Press Play

Scratch

```
ask First Number and wait
set no1 to answer
ask Second Number and wait
set no2 to answer
say no1 + no2
```



EduBlocks

```
no1 = int(input("First Number"))
no2 = int(input("Second Number"))
print(no1 + no2)
```

7

Python

```
no1 = int(input("First Number"))
no2 = int(input("Second Number"))
print(no1 + no2)
```

7

How to run your code:

Scratch



Click the first block

EduBlocks



Click the Run Button

Python



Press Play

1 Scratch

```
define my_function
```

```
say Hello from a function!
```

```
my_function
```



Hello from a function!

2 EduBlocks

```
def my_function ( )::
```

```
print(" Hello from a function! ")
```

```
my_function ( )
```

Hello from a function!

3 Python

```
def my_function():  
    print("Hello from a function!")  
my_function()
```

Hello from a function!

How to run your code:

Scratch



Click the first block

EduBlocks



Click the Run Button

Python



Press Play

123 Coding Cards

9: 10 Second Timer

Scratch

```
repeat 10
  change timer by 1
  wait 1 seconds
  say timer
```



EduBlocks

```
import time
for i in range(10):
  print(i)
  time.sleep(1)
```

0 5
1 6
2 7
3 8
4 9

Python

```
import time
for i in range(10):
  print(i)
  time.sleep(1)
```

0 5
1 6
2 7
3 8
4 9

How to run your code:

Scratch



Click the first block

EduBlocks



Click the Run Button

Python



Press Play

Scratch

```
ask "What's your name?" and wait
repeat 2
  say "Happy Birthday To You"
  wait 1 seconds
say join "Happy Birthday Dear" answer
wait 1 seconds
say "Happy Birthday To You"
```



Happy Birthday To You

EduBlocks

```
import time
name = input("What is your name?")
for i in range( 2 ):
  print(" Happy Birthday To You ")
  time.sleep( 1 )
print("Happy Birthday dear " + name )
time.sleep( 1 )
print(" Happy Birthday To You ")
```

What is your name? **George**
Happy Birthday To You
Happy Birthday To You
Happy Birthday Dear **George**
Happy Birthday To You

Python

```
import time
name = input("What is your name?")
for i in range(2):
  print("Happy Birthday To You")
  time.sleep(1)
print("Happy Birthday dear " + name)
time.sleep(1)
print("Happy Birthday To You")
```

What is your name? **George**
Happy Birthday To You
Happy Birthday To You
Happy Birthday Dear **George**
Happy Birthday To You

How to run your code:

Scratch



Click the first block

EduBlocks



Click the Run Button

Python



Press Play